23-07-93 13:39 FROM CULHAM LABS ID 0235 463480 P. 2

(1)

AD-A269 085

[illegible]NTATION PAGE OMB No. 0704-0188

Report Date: June 1993	3. Report Type and Dates Covered: Final Report
4. Title and Subtitle: General Geometry PIC for MIMD Computers: Final Report	5. Funding Numbers: F 49620-92-C-0035
6. Author(s): J W Eastwood, W Arter and R W Hockney	
7. Performing Organization Name(s) and Address(es): Culham Laboratory, Radio Frequency Effects, Abingdon OX 14 3DB, UK	8. Performing Organization Report Number: AEA/TLNA/31858/RP/2
9. Sponsoring/Monitoring Agency Name(s) and Address(es): Sponsoring Agency: Phillips Laboratory, Kirtland AFB NM 87117-6008; Sponsoring/Monitoring Agency: European Office of Aerospace Research and Development, PSC 802 Box 14, FPO AE 09499-0200	10. Sponsoring/Monitoring Agency Report Number: 93 [illegible]
11. Supplementary Notes:	
12a. Distribution/Availability Statement: Approved for Public Release; Distribution Unlimited	12b. Distribution Code:

13. Abstract (Maximum 200 words)

New relativistic PIC algorithms using body fitted for modelling microwave sources in two and three dimensions have been derived. A parallel computer benchmarking program using a multiblock implementation of the two dimensional algorithms has been written. Test computations performed on the iPSC computer at Phillips Laboratory have demonstrated the MIMD computers can be efficiently used to model devices with complex geometry.

14. Subject Terms (Key Words) PIC, Electromagnetics, Parallel Computers			15. Number of Pages
			16. Price Code
17. Security Classification of Report: UNCLASSIFIED	18. Security Classification of This Page: UNCLASSIFIED	19. Security Classification of Abstract: UNCLASSIFIED	20. Limitation of Abstract: UNCLASSIFIED

RFFX(93)56

COMMERCIAL IN CONFIDENCE

GENERAL GEOMETRY PIC FOR MIMD COMPUTERS: FINAL REPORT

James W EASTWOOD
Roger W HOCKNEY
Wayne ARTER

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB
England

June 1993

Document Control Number: AEA/TLNA/31858/RP/2			
Date of Issue: July 8, 1993			Issue number: 1
Authorization	*Name*	*Signature*	*Position*
Prepared by	J.W. Eastwood	[illegible]	Project Manager
Approved by	D.E.T.F Ashby	[illegible]	Department Head

CONTENTS

GENERAL GEOMETRY PIC FOR MIMD COMPUTERS: FINAL REPORT

James W Eastwood, Wayne Arter and Roger W Hockney

June 1993

SUMMARY

The objectives of the work programme specified in the Proposal[3], namely the

(i) derivation of MIMD oriented algorithms in general curvilinears,

(ii) development of a 2-D multiblock benchmarking computer program, and

(iii) execution of benchmarking computations

have all been achieved. The conclusion from the study is that the proposed methods will efficiently use MIMD computers in the design and evaluation of HPM sources with complex geometries. When implemented on a large parallel computer, the general geometry PIC schemes will allow hitherto unattainable accuracies and system sizes to be modelled.

1 Introduction

The work described in this Final Report summarises the work undertaken in the research project sponsored by the Air Force Office of Scientific Research (AFSC) under Contract F49620-92-C-0035, "General Geometry PIC Algorithms and Software for Distributed Memory MIMD Computers". The objectives of the work programme of this Contract are described in the proposal [3]

J W Eastwood, *A Proposal to Develop General geometry PIC Algorithms and Software for Distributed Memory MIMD Computers*, RP363, Culham Laboratory, Nov 1991.

These objectives may be summarised as

- the derivation of MIMD orientated algorithms in general curvilinears,
- the development of a 2-D multiblock benchmarking computer program,
- benchmarking computations.

The approach proposed, and successfully implemented uses

- the 'Virtual Particle' derivation method [2] applied to tensor field components,
- a multiblock spatial decomposition applied to both fields and particles,
- transfinite interpolation subdivision of the curvilinear quadrilateral multiblocks into quadrilateral elements,
- indirect ("glue patch") addressing between multiblocks, and logical square mesh (i,j) node addressing within blocks.

The programme of work was divided into four Tasks:

Task 1 Prepare a Report containing the derivation and statement of the *general geometry electromagnetic particle in cell algorithms* to be implemented in the 2-D benchmark program.

Task 2 Develop the single block solver software modules for:

(i) explicit time stepping of Maxwell's equations,

(ii) explicit particle time stepping,

(iii) glue patch transformations.

Task 3 Develop the 2-D multiblock benchmarking program.

Task 4 Perform demonstration test runs on the Intel computer at Phillips Laboratory.

These Tasks and their outcome are described in detail in the following four Sections.

In addition to the Final Report, Culham undertook to supply (subject to IPR conditions specified in [3]) a final version of the software and test input and output data sets. These have already been delivered and installed on computers at Phillips Laboratory. Conclusions and recommendations from the work undertaken on this Contract are described in the final Section.

Whilst the immediate goals of the work on this Contract are Tasks 1-4 listed above, a broader view has been taken in treating them as steps towards the ultimate practical realisation of state-of-the-art simulation software for three dimensional configurations. The software is intended primarily to aid in the design and interpretation of HPM sources and power transmission systems, although the fundamental nature of the core software is such that it may be of utility in other computational electromagnetic applications.

2 Task 1: Algorithm Derivation

The physics, numerical analysis, software design, the impact of contemporary computer architecture, and the eventual need to extend to three dimensions have all been taken into account in the algorithm derivation and in the design and implementation of the software. The approach adopted is that which we believe most effectively realises the underlying aim of the work, and has the following features:

- a variational finite element derivation, using tensor fields in the action integral formulation of Maxwell's equations,
- covariant (E and H) and contravariant (d and b) electromagnetic field components,
- orthogonal coordinates where appropriate, but general curvilinear coordinates where the extra freedom is needed,
- multiblock spatial decomposition of complex domains,
- indirect ("glue patch") addressing between blocks,
- regular (i, j, k) cubic lattice addressing within blocks,
- transfinite interpolation subdivision of curvilinear hexahedral blocks into finite elements,
- compact metric storage, and
- data and algorithm organisation optimised to exploit distributed memory MIMD computers.

To design software, which not only meets the objectives (see above) but also provides the basis for a general curvilinear field solver with multiblock decomposition amenable to distributed memory MIMD implementation, has demanded extensive numerical analysis, software engineering and benchmarking. In the course of this work, a number of problems have been encountered; for example, in discretising the constitutive relationships and boundary conditions, in defining an effective data storage scheme for handling the metric special cases, in designing for high computational intensity in MIMD multiblock implementations, and so forth. All problems encountered have been substantially overcome.

A detailed discussion of the algorithm derivation and statement is given in the Task 1 Report

J W Eastwood, R W Hockney and W Arter, *General Geometry PIC for MIMD Computers* RFFX(92)52, Culham Laboratory, August 1992

which is appended to the present Report as Annex 1.

3 Task 2: Uniblock Solver Software

The general geometry VP electromagnetic PIC derivation described in Annex 1 reduces the problem of modelling a complex shaped device into that of computing fields and particle dynamics in a set of rectangular blocks (in curved space) with boundary conditions applied only at the surfaces of the blocks. Boundary conditions are implemented by copying data from the source block surfaces (currents for Ampere's Law, electric fields for Faraday's Law and particle coordinates for the equations of motion) to the source glue patch buffers, performing appropriate transformation of data in the gluepatch buffers, and then copying the results from the target gluepatch buffers to the target blocks.

Each uniblock of the multiblock decomposition behaves as an independent computation with boundary conditions provided by surface boundary condition patches or gluepatches. This independence is reflected in the coding of the uniblock subprograms, which are written in terms of the local block indexing and local position coordinates.

The objective of Task 2 was to write Fortran code for the uniblock calculations. These subprograms may be divided as follows:

(i) electromagnetic routines

(ii) electromagnetic boundary condition routines

(iii) particle routines

(iv) particle boundary condition routines.

Annex 2 lists the documentation modules from the Fortran benchmark program delivered to Phillips Laboratory. The uniblock subprograms described below are provided as part of the benchmark program. The OLYMPUS conventions for notation, layout and documentation [1] have been followed. Subprograms with the same names as those in the prototypical timestepping program CRONUS have the same function as the CRONUS dummy counterparts.

In referring to subprograms in the following subsection the notation ⟨c.s⟩name will be used, so for example ⟨2.20⟩ AMPERE will refer to the main calculation (class 2) routine, subroutine number 20, which is stored in file c2s20.f.

The contents of file c2s20.f will be subroutine **AMPERE**, which computes the displacement current (cf documentation file **INDSUB**.doc in Annex 2). Similarly, common blocks have their group and member numbers; for example, **[C4.5] COMADP** is a member of group 4 (housekeeping) blocks with name **COMADP**, etc.

3.1 Electromagnetic routines

The principal routines for advancing Maxwell's equation in each uniblock are

⟨2.20⟩ **AMPERE** which computes scaled contravariant displacement currents, d^i, according to eq(4.6.5) of Annex 1.

⟨2.21⟩ **FARADA** which updates scaled contravariant magnetic fields, b^i, according to eq(4.6.8) of Annex 1.

⟨2.22⟩ **GBTOH** which computes covariant magnetic field intensities, H_i, from b^i (eq(4.5.1) of Annex 1).

⟨2.23⟩ **GDTOH** which computes covariant electric fields, E_i, from d^i (eq(4.5.2) of Annex 1).

Input and output to these routines is via formal parameters. Included common blocks are **[C1.9] COMDDP** and **[C4.5] COMADP**. The first of these is to give access to OLYMPUS development and diagnostic parameters for code testing, and the second contains symbolic names of addressing constants. The use of symbolic addressing constants was adopted to allow dimensionality and data storage layout to be changed without recoding the uniblock routines.

The remaining uniblock electromagnetic routines are general mesh manipulation routines, applicable to any vector field defined on the element net. These routines perform the following functions.

⟨1.21⟩ **NILVEC** sets all nodal amplitudes of a vector field on the element net to zero

⟨1.20⟩ **SETVEC** sets nodal amplitudes to a constant vector

⟨2.26⟩ **ADDVEC** adds two vector fields together

⟨2.34⟩ **CPYVEC** copies one vector field to another

⟨2.36⟩ **AVEVEC** averages two vector fields.

Specifications of the input to and output from these electromagnetic routines of the uniblock solver software are given in the nine subsections of Appendix A.1. These routines can be used as they stand for both two and three dimensional calculations.

3.2 Electromagnetic boundary condition routines

The uniblock electromagnetic boundary condition routines (2.25) BCOPAT and (2.28) BCONE are used respectively to apply external boundary conditions on the displacement field and electric field. The present versions of these routines can apply conductor, applied field and isotropic resistive wall boundary conditions for element nets which are orthogonal at the external boundaries. The routines are written to work in both two and three dimensions.

Internal boundary conditions between blocks are handled by the gluepatch routine (2.30) GLUEIO. This routine works for othogonal and non-orthogonal in both two and three dimensions.

A specification of the input to and output from the uniblock electromagnetic boundary condition routines listed above is given in the Appendix, subsection A.2.

3.3 Particle routines

The principal particle integration routines are

(2.11)MOVCUR , which updates the particle position coordinates and assigns current to the element net according to eqs(5.2.9)-(5.2.12) of Annex 1

(2.12)ACCEL , which updates particle momenta using the scheme described in Section 5.4.2 of Annex 1.

MOVCUR calls subsidiary routine MERGE in computing the location of the virtual particles, and calls ASSCUR to assign current from the particles to the element net.

A specification of the input to and output from these uniblock particle routines is given in the Appendix, subsection A.3. The first subprogram listed therein, (2.10) SETCUR reinitialises the current accumulation arrays each timestep.

3.4 Particle boundary condition routines

Internal particle boundary conditions and particle absorption at external boundaries are handled by the gluepatch routine (2.14)PARTIO of the multiblock test program (cf Section 3). Particle injection is handled by (2.19) INJECT, which uses the uniblock routine (2.18)QSHARE to find charge density at cathode surfaces and (2.13)EMITEL to emit electrons from the cathode surface using a space charge limited emission algorithm.

Table 1: *Global Addressing Arrays*

Name	Meaning
MPESTB	Processor to process pointer table
MPORTB	Process to processor pointer table
MBKPES	Block to process pointer table
MPRBLK/NXTBLK	Process to block pointer header and link tables
MPATBK	Patch to block pointer table
MBKPAT/NXTPAT	Patch to block pointer header and link tables
MBKTYP	Block to blocktype pointer table
MBCPBK	Boundary condition (bc) patch to block pointer table
MBKBCP/NXTBCP	Block to bc patch pointer header and link tables
MPATBK	Gluepatch to block pointer table
MBKPAT/NXTPAT	Block to gluepatch pointer header and link tables

A specification of the input to and output from these uniblock particle boundary condition routines is given in the Appendix, subsection A.4.

4 Task 3: Multiblock Test Program

The Multiblock Test Program provides the testbed in which the uniblock modules described in the previous section are combined with control and message passing routines to form the MIMD-PIC test program.

The main control routine which calls the various uniblock routines is (2.1)STEPON. STEPON calls further routines (2.24)BCSURD, (2.26)BCSURE and (2.29)BCSYM to control the application of external electromagnetic boundary conditions, and (2.32)BLKIO to handle the transformation and copying of electromagnetic gluepatch data to and from the gluepatch buffer arrays. Particle data is copied to and from gluepatch buffer arrays by (2.14)PARTIO. Particle injection boundary conditions are controlled by (2.19)INJECT.

4.1 Data Organisation

Global and local data organisation in the program both follow the scheme outlined in Annex 1. Table 1 summarises the array names for pointer tables connecting processor to process, process to block, block to blocktype, block to boundary patch and block to gluepatch. In most cases there are two-way pointers, and in instances where two names are given, pointers are implemented as linked lists.

Global data have the same values on all processes. Local data have different values on different processes. Local data includes the gluepatch to buffer pointers (MGLTOB and MBTOGL), field and particle addressing data, and the field and coordinate values.

Field data for each uniblock is mapped onto one dimensional Fortran arrays as described in Section 7.4 of Annex 1. Uniblock subprograms (cf Section 3) are written relative to origin 1 in the field and addressing arrays. The relevant origin location in the multiblock arrays *for each block is passed* to the subprograms by calling with the appropriate offsets. This can be seen by inspection of (2.1)STEPON. For example

```
C
CL                1.2      clear current arrays
          CALL SETCUR(
     +    LBLAS (LOBLAS(IBTYPE)),
     +    C     (LORFBL(IBLOCK)) )
C
```

passes the block addressing information in array LBLAS for block type IBTYPE, and sets currents C to zero for block IBLOCK.

Similarly, particle address and coordinates are stored in one dimensional arrays LPARAS and COORDS, respectively. Offsets for each block are passed *to uniblock routines* by calling with the pointer to the location of origins of coordinate (LOCOOR) to the current block, for example

```
C
C    move
          CALL MOVCUR(0,
     +    LPARAS(LOPARA(IBLOCK)),
     +    SPATR,
     +    LECOVA(1,LOECOA(IBTYPE)),
     +    ECOV(LOECOV(IBTYPE)),
     +    COORDS(LOCOOR(IBLOCK)),
     +    GPATI,
     +    GPATI,
     +    LBLAS (LOBLAS(IBTYPE)),
     +    C     (LORFBL(IBLOCK)) )
C
```

For further information on the data storage, see Annex 2 and the documented program listing.

4.1.1 Program Structure

The benchmark program, MIMD-PIC, follows the canonical notation and structure of the skeleton initial value/boundary value timestepping program CRONUS [1]. Program flow is controlled by ⟨0.3⟩COTROL, the main timestep loop is controlled by ⟨2.1⟩STEPON and output is controlled by ⟨3.1⟩OUTPUT.

The decimal numbered subsections of ⟨2.1⟩STEPON reflect the steps of the timestep loop:-

```
(1) move particles and compute currents
    loop blocks or process (1.1)
        (1.2) initialise current (SETCUR)
        (1.3) inject particles (INJECT)
        (1.4) move and accumulate current (MOVCUR)
        (1.5) sort lost particles to gluepatch buffer (PARTIO)
    end loop
    loop to empty gluepatch buffers
        (1.6) exchange particles (XPART)
        (1.7) complete move for exchanged particles
        loop blocks on process
            move particle from buffer to block (MOVCUR)
            sort lost particles to gluepatch buffer (PARTIO)
        end loop
    end loop

(2) update displacement field
    loop blocks on process
        (2.1) compute $H$ from $b$ (GBTOH)
        (2.2) compute displacement current (AMPERE)
        (2.3) load gluepatch buffers (BLKIO)
    end loop
    (2.4) exchange gluepatch buffers (XPATCH)
    loop blocks on process
        (2.5) copy gluepatches to blocks (BLKIO)
    end loop
    (2.6) apply boundary conditions to $d$ (BCSYM, BCSURD)

(3) compute new $E$ field
    loop blocks on process
        (3.1) compute $E$ from $d$ (GDTOE)
        (3.2) load gluepatch buffers (BLKIO)
    end loop
    (3.3) exchange gluepatch buffers (XPATCH)
    loop blocks on process
        (3.4) copy glue patches to blocks (BLKIO)
    end loop
    (3.5) apply boundary conditions to $E$ (BCSYM, BCSURE)

(4) advance magnetic fields
    loop blocks on process
        save old $b$ (CPYVEC)
        advance $b$ (FARADA)
        compute time centred $b$ (AVEVEC)

(5) accelerate particles
        ACCEL
    end loop
```

The numbering in this summary of the timestep loop corresponds to the section numbers in (2.1) STEPON, and the names in brackets are the subprogram names.

4.2 Test Cases

The source, executables and test data for the workstation version of MIMD-PIC has been installed in directory *may12.d* on the SUN workstation *ppws04* at Phillips Laboratory. This version differs primarily from the iPSC version in that calls to Intel interprocessor communications routines have been replaced by dummies. To execute the program using one of the test datasets, e.g. *test17.dat* type

```
xmimdpic test17.dat
```

If the *xghost* library has been linked into *xmimdpic*, then a window with graphics output will appear, otherwise only printer output file *o_test17p1*, restart binary file *r_test17p1* and graphics output file *g_test17p1* will be produced. The graphics file may be viewed using the GHOST interactive viewer program *xghost*[5].

The test data sets included in the directory *may12.d* are

test1.dat	:	coded exchange test
test2.dat	:	current and d array
test3.dat	:	transmission line test/constant d
test4.dat	:	transmission line test/travelling wave

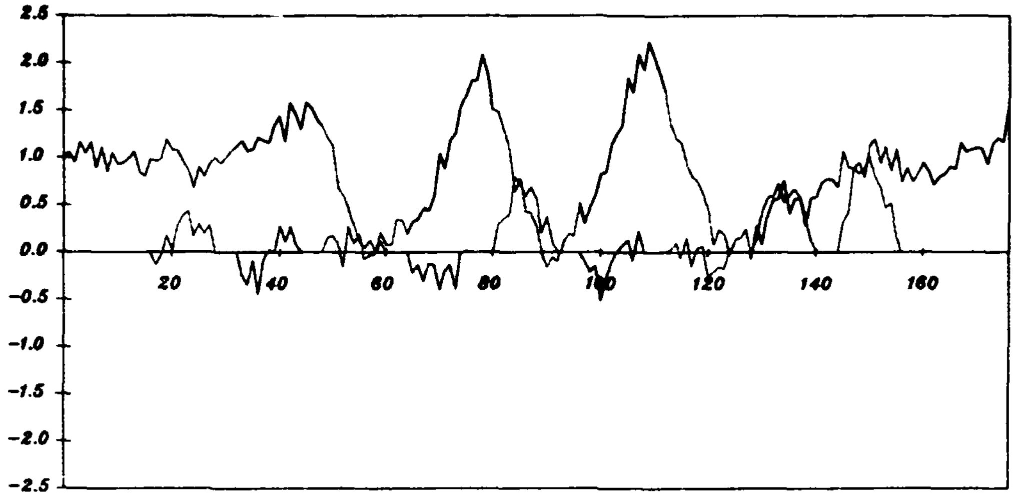
test5.dat	:	3-D coded current array
test6.dat	:	3-D field test
test7.dat	:	particle mover test
test8.dat	:	particle mover test
test9.dat	:	periodic mover test
test10.dat	:	cyclotron orbit
test11.dat	:	cyclotron with E × B
test12.dat	:	current assignment check
test13.dat	:	short MITL test
test14.dat	:	longer MITL
test15.dat	:	Further E × B test
test16.dat	:	1 cavity device/no particle
test17.dat	:	5 cavity MILO/few particles
test18.dat	:	4 cavity MILO/50 steps
test19.dat	:	Symmetry bc EM transmission line
test20.dat	:	test error in G2LMAT
test21.dat	:	particle reflection bc test
test22.dat	:	uniform *d* start 5 cavity milo/sheat plot
test23.dat	:	1 block MILO test
test24.dat	:	20 block MILO test/5000 steps
test25.dat	:	20 block/finer mesh MILO 15,000 steps
test26.dat	:	04 block MILO
test31.dat	:	3-D version of test 1
test33.dat	:	3-D version of test 3
test34.dat	:	3-D version of test 4

The input and output dataset for these test cases are in directory *may12.d* under user *eastwood* on the *ppws04* machine. input dataset *testnn.dat* has a corresponding output file *o_testnnp1*, GHOST graphical output file *g_testnnp1* and restart file *r_testnnp1*. (The suffix 'p1' is for a one processor run. Output from m processor runs on the iPSC have suffices pm:r for the rth processor output from an m processor hypercube).

The input and output datasets are largely self explanatory, so will not be described further here. The graphical output files may be viewed either using the GHOST interactive X-window viewer *xghost*, or by converting them to (say) Postscript files and printing them.

5 Task 4: iPSC Parallel Benchmarking

The parallel implementation of the multiblock program differs only in the handling of the gluepatch buffer exchange between uniblocks. In the serial code, gluepatch exchange involves only memory to memory copying.



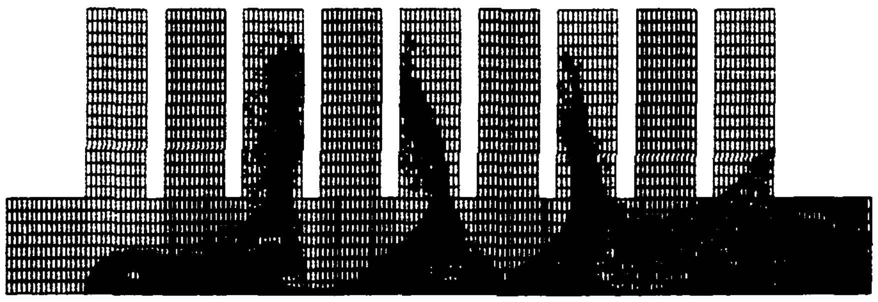


Figure 1: *A snapshot of the radial fields and electron distributionsfrom the MILO test run test25. The electric fields are measured along lines parallel to the axis of the device through the centre of the drift space (green/black curve), and half way up the cavities (red/blue curve). Fields are in units of applied field, and distance is in element widths.*

The parallel code uses memory to memory copying for gluepatches between uniblocks on the same processor, and uses interprocessor message passing between blocks on different processors. The gluepatch exchange subprogram (2.31)XPATCH contains explicit calls to iPSC routines, and in the serial version, these routines are replaced by dummies (cf file *cps1.f*).

The test cases described below were chosen firstly to show that the parallel implementation worked correctly, and secondly to evaluate the performance enhancements that could be achieved by parallel processing.

5.1 Test problems

The test problems for the parallel benchmarking were chosen subject to the constraints that

- they use simple geometrical elements, since the general metric element computation routines have not been incorporated in the benchmark software,
- they reflect realistic engineering types of calculations where the domains are not simply rectangles, and the particle filling is nonuniform,
- the results can be cross checked with those from existing serial codes.

These constraints led to a planar MILO configuration being chosen for the example geometry.

Typically, MILO calculations are started from an empty device with zero fields, and an electric field is applied at the generator boundary. Computations are then run for several tens of thousands of step. Figure 1 shows the axial electric fields and electron distribution for one such calculation after 12,000 steps. This computation showed the same behaviour as a cross check using the MAGIC code [6]

The two test cases for the parallel computations are the same as two of the test cases listed in Table 1. The input datasets *case1.dat* and *case2.dat* are the parallel computation equivalents of *test17.dat* and *test24.dat*, respectively. The datasets for the parallel runs differ from the serial ones only in the addition of an extra data input variable to specify the number of processors to be used for the computation.

case1.dat (test17.dat) is a small calculation example, which describes a five cavity MILO using 12 uniblocks with coarse element nets. The 100 step run follows the standard MILO startup from an empty field-free configuration. Large electron superparticles are used so that by the end of the 100 step run there are less than 140 particle in the MILO. Running the profiler

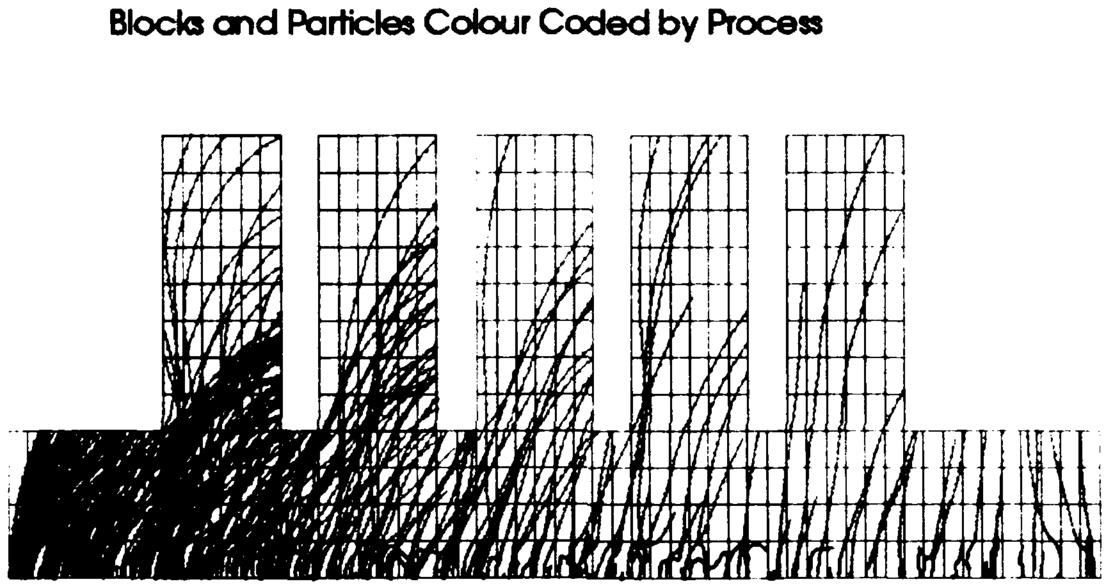


Figure 2: *The streak plot of particle trajectories for iPSC run using dataset case1.dat shows the correct passing of particle coordinates between uniblocks*

on the workstation version of the code for this test case showed that the mix of work was uncharacteristic of particle calculations. The breakdown of the main elements of the timestep cycle was as follows:

34%	PARTIO and BLKIO
25%	AMPERE, GDTOE, FARADA and GBTOH
14%	MOVCUR and ACCEL

The dominant part of the calculation cycle is the work to exchange data between block, followed by the work in the electromagnetic field calculation.

case2.dat (test24.dat) is a medium sized calculation. It was chosen to reflect the general characteristics of a production calculation, but in a run which takes only a few (100) timesteps. This differs from the normal MILO runs in that the device is initially filled with a nonzero electric and magnetic field. The result is that electrons are emitted from the whole length of the cathode, and there is a strong initial transient where the electrons fill only part of the device volume. The percentages of the calculation time taken by the patch exchange, electromagnetic and particle parts are now

6%	PARTIO and BLKIO
12%	AMPERE, GDTOE, FARADA and GBTOH
73%	MOVCUR and ACCEL

This ordering of the amount of work is more typical of realistic particle computations, where the particle integration dominates. The interblock data transfer routines are now a small part of the serial calculation. Since it is only this part of the calculation which involves message passing in the parallel implementation the scope for parallel speedup for *case2.dat* will be much greater than for *case1*.

Further cases, where the benchmark code is extended to undertake three dimensional calculations, are planned for the future. On the basis of computational intensity estimates, we anticipate much greater scope for speedup than can be achieved in two dimensions. 3-D tests would be invaluable in evaluating the capabilities of machines such as the Intel Paragon for use in large scale electromagnetic computations.

5.2 Benchmark results

Case-1 has been used as a validation example throughout the development of the parallel code, because it contains particles which pass through several blocks belonging to different processors. To demonstrate that these complicated cases are being correctly computed, Fig. 2 shows the orbits of particles in which the colour changes as the particles are computed by different processors. Such a colour change indicates that the particle coordinates are being correctly transferred by way of a message containing their coordinates being sent from one processor to another. We also note that the orbits do not stop at the non-physical boundary between the regions computed by different processors, as would be the case if the message transfer between processors was not operating correctly.

Figure 3 shows the scaling charcteristics of case-1 when run on an iPSC/860 with up to 16 processors, together with the ideal speedup line. The unit of performance used is timesteps per second (tstep/s), and the graph shows how the performance scales for a fixed size problem as the number of processors increases. This unit of performance is to be preferred to the traditional Speedup (ratio of p-processor performance to one processor performance), because it retains the absolute speed of the calculation which is of more interest to the user than a speedup ratio. The ideal linear speedup line is also shown for the case with output to the cube disk and additional diagnostic output switched on (NLREPT=T); this is the theoretical performance which would be obtained if the performance scaled up linearly with the number of processors. This curve will not be reached in practice because of load imbalance, communication overheads, and essentially serial code that cannot be parallelised. The latter includes any code that is repeated for convenience in all processors, and will eventually cause the performance to saturate (Amdahl saturation) and even subsequently decrease with the number of processors.

Three curves are shown depending on whether the printed output is returned to the host (bottom two curves marked to HOST), or whether it is sent to the to the disks that are directly connected to the iPSC/860 hypercube (top curve marked to CUBE). The output is controlled by the variable NLREPT=T (full report) or NLREPT=F (minimal report). As already noted, case-1 is a small test calculation used in program development, and is too small to make efficient use of a large MPP. It is not surprising, therefore, that the performance scaling is poor. All curves show the onset of Amdahl saturation, due to repeating the output in all processors, but the importance of using the CUBE, rather than HOST file system is evident.

Figure 4 shows the scaling behaviour for the medium sized case-2, compiled under both if77 (UNOPTIMISED) and if77 -O (OPTIMISED). The ideal linear speedup is fitted to the optimised case. The gain obtained using compiler optimisation is clearly seen. Although the performance can be seen to fall-off from the linear speedup line, the scaling can be regarded as reasonable for a problem of this size. Even when there are only four blocks per processor, more than 75 per cent of the ideal linear speedup is realised. Unless the uniblock are further subdivided, saturation for this case must be reached by 64 processors. However, for the type of calculation for which this

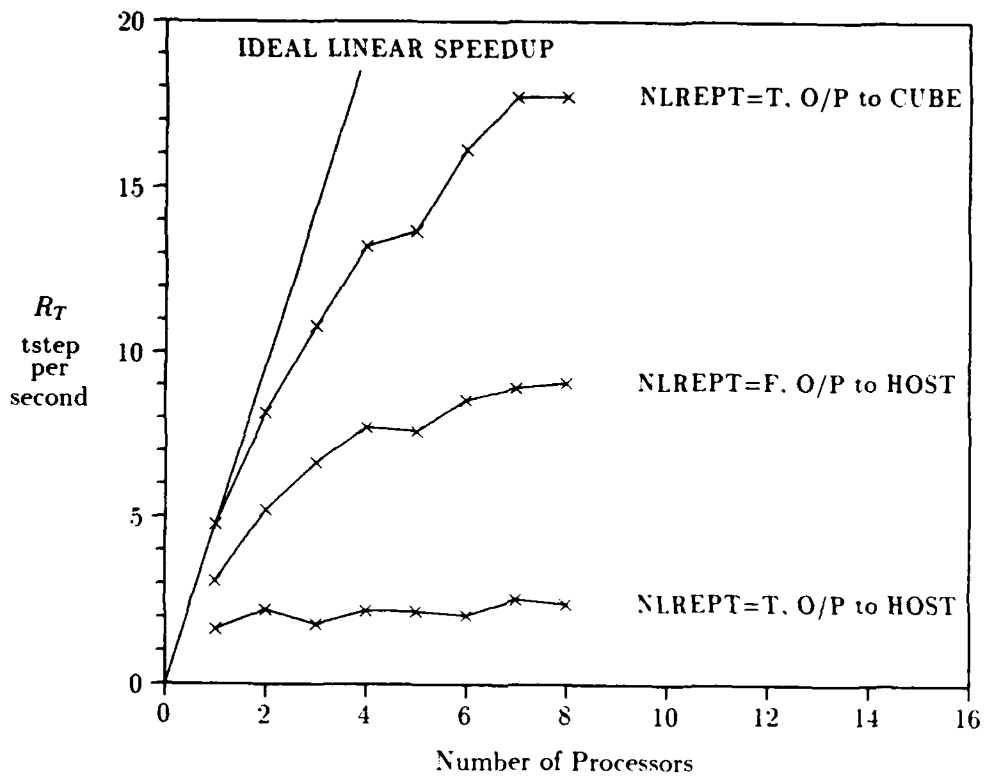


Figure 3: Temporal Performance in timestep per second (tstep/s) for case-1.

software is designed, we would not expect to meet performance saturation until several hundred processors were used. More data is required is required for the larger cases to determine the best buffering strategy, and adapt the software to return the best performance. This requires access to an Intel Paragon with at least 200 processors, and preferably more.

6 Final Remarks

The objectives of the work programme specified in the Proposal[3], namely the

(i) derivation of MIMD oriented algorithms in general curvilinears,

(ii) development of a 2-D multiblock benchmarking computer program, and

(iii) execution of benchmarking computations

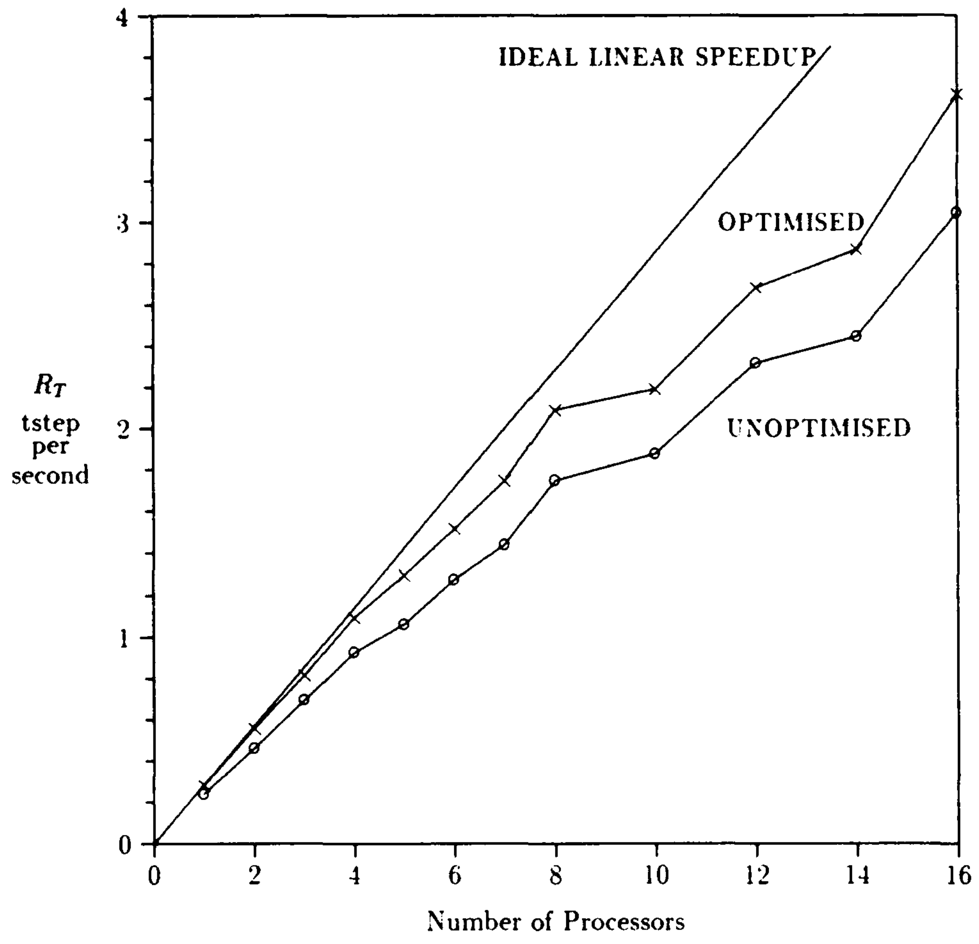


Figure 4: Temporal Performance in timestep per second (tstep/s) for case-2.

have all been achieved. In some aspects the results achieved have exceeded requirements. For instance, the uniblock routines (Section 3) and the multiblock test program (Section 4) can already be used in their present form for both two and three dimensional cases. Although the software is suitable for parallel benchmarking, much work remains before it becomes a usable tool for microwave device simulation; the initialisation is difficult to use in its present form, and still lacks the subprograms to compute metric tensor elements for general curvilinear uniblocks; also the output from the code is quite limited. Further development and a considerable amount of validation is required before the software can be regarded as sufficiently debugged for routine microwave computations.

Most of the core timestepping routines for the three dimensional extension of the benchmarking code are complete. Known exceptions to this are

(i) the ability to handle general curvilinear external boundary conditions on the electromagnetic fields.

(ii) non-lumped approximations to the electromagnetic equations.

(iii) three dimensional space charge limited and beam emission particle boundary conditions.

It emerged during the implementation of the particle integration routines that a more efficient particle momentum integration may result from using local non-orthogonal coordinates rather than local cartesians; we recommend that the question as to which approach is most effective is resolved before further extension of the particle software is undertaken.

The results of the preliminary benchmark computations showed encouraging speed up, even for modestly sized calculations. The present implementation bases message passing on a patch to patch basis. On machines with high interprocessor message passing latency, further speed up would result from presorting the patches and performing message passing on a processor by processor basis (cf below).

6.1 The LPM2 benchmark

The results reported in section 5.2 show that the benchmark version of the new parallel code works, and can be used to test the scaling behaviour of Massively Parallel Processors (MPPs) that may be considered for acquisition by the USAF, both now and in the future. The most useful way of doing this is to offer the benchmark (which we have called LPM2 for Local-Particle-Mesh #2) as a component of a widely disseminated benchmark set; this will result in performance numbers being produced by the manufacturers as a matter of course on all their new computers. Thus the USAF would see that performance of one of their important class of codes quoted without having to take any action, much in the same way that LINPACK benchmark results currently appear.

A recent initiative by large scale computer users was taken at Supercomputing92 for exactly this purpose and has held three meetings. This committee, called the ParkBench committee (Parallel Kernel Benchmarks), aims to specify a set of public-domain benchmarks for the evaluation of parallel systems and MPPs that is acceptable to both the user community and the manufacturers. This committee is currently chaired by Professor Hockney, and is looking for a parallelised PIC code, which expands up to MPP sized problems. In our view it would be in the long-term advantage of the USAF to put the LPM2 benchmark code into the public domain and offer it to fulfill this role in the ParkBench benchmark suite. If USAF is agreeable to this action Professor Hockney will undertake to submit LPM2 to ParkBench and hopefully have it encorporated in the benchmark suite.

In order to test the behaviour of the new parallel code fully it is necessary to run it on a real machine with several hundred processors. This should be a Paragon, and/or competitive computer (e.g. Meiko CS2, currently being considered by LLL). This will determine the extent of performance saturation which occurs with the present code. There are steps that can then be taken to improve the performance at saturation, and to delay the onset of saturation. However without access to a large MPP, the timings necessary for such an optimisation cannot be performed. We place high priority on obtaining such measurements as the first stage of the proposed future work.

Early measurements on the Intel Paragon [7] show that, although the asymptotic bandwidth of the Paragon is about eight times that of the iPSC/860, the message startup time is about twice as long as that of the Intel iPSC/860. Since the arithmetic processors used on the two computers are the same, the Paragon will show improved performance over the iPSC/860 only if a few long rather than many short messages are sent. This puts a premium on merging many small messages into one large one and then sorting the results after the message has arrived. The current code does not exploit this possibility, and we recommend that one line of future development be to insert such message merging and sorting code.

7 References

[1] R W Hockney and J W Eastwood, *Computer Simulation Using Particles*, (Adam Hilger/IOP Publishing, Bristol & New York, 1988).

[2] J W Eastwood, The Virtual Particle Electromagnetic Particle-Mesh Method, *Computer Phys Commun* **64**(1991)252-266.

[3] J W Eastwood, *A Proposal to Develop General Geometry PIC Algorithms and Software for Distributed Memory MIMD Computers*, Culham Laboratory Proposal RP363, Nov 1991.

[4] J W Eastwood, R W Hockney and W Arter, *General geometry PIC for MIMD Computers*, Report RFFX(92)52, Culham Laboratory, August 1992.

[5] W A J Prior, GHOST User Manual Version 8, UKAEA Culham Laboratory, 1991.

[6] M Amman, *private communication*

[7] R W Hockney, *private communication*

A Appendix: Uniblock Subroutine Input and Output Specification

A.1 Electromagnetic routines

A.1.1 Subroutine AMPERE

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

C	*Real array*	contravariant current
H	*Real array*	covariant magnetic intensity
KBLAS	*Integer array*	block addressing structure

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

DDOT	*Real array*	contravariant displacement current

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.2 Subroutine FARADA

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

B	*Real array*	scaled contravariant magnetic field
E	*Real array*	covariant electric field
KBLAS	*Integer array*	block addressing structure

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

B	*Real array*	scaled contravariant magnetic field

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.3 Subroutine GBTOH

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

B	*Real array*	scaled contravariant magnetic field
GBH	*Real array*	b^i to H_i conversion tensor
KBLAS	*Integer array*	block addressing structure
KGBHAD	*Integer array*	GBH addressing structure

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MINCOG	*Integer*	Mesh INCrement Origin in G addressing
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

H	*Real array*	covariant magnetic intensity

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.4 Subroutine GDTOE

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

D	*Real array*	scaled contravariant displacement field
GED	*Real array*	d^i to E_i conversion tensor
KBLAS	*Integer array*	block addressing structure
KGEDAD	*Integer array*	GED addressing structure

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MINCOG	*Integer*	Mesh INCrement Origin in G addressing
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MORIGG	*Integer*	Mesh ORIGin in G addressing
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

B	*Real array*	scaled contravariant magnetic field

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.5 Subroutine NILVEC

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MORIGG	*Integer*	Mesh ORIGin in G addressing
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT1	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PV	*Real*	vector field set to zero

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.6 Subroutine SETVEC

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
KCASE	*Integer*	=1 for *d* and =2 for *b* fields
PVALS	*Real array*	3- vector of values to which PV is set

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT1	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PV	*Real array*	vector to be set to PVALS

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.7 Subroutine ADDVEC

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
PV1	*Real array*	input vector 1
PV2	*Real array*	input vector 2

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PV1	*Real array*	output vector PV1 := PV1 + PV2

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.8 Subroutine CPYVEC

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
PV2	*Real array*	input vector

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh othogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PV1	*Real array*	output vector set to PV2

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.1.9 Subroutine AVEVEC

External Routines

EXPERT

Intrinsic Functions

MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
PV1	*Real array*	input vector 1
PV2	*Real array*	input vector 2

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PV1	*Real array*	output PV1 := (PV1 + PV2)/2

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.2 Electromagnetic boundary condition routines

A.2.1 Subroutine BCOPAT

External Routines

EXPERT
MESAGE

Intrinsic Functions

MAX
MOD

Arguments on Entry

D	*Real array*	displacement field
DDOT	*Real array*	displacement current
KBLAS	*Integer array*	block addressing structure
KO	*Integer array*	location of patch origin
KPBCAT	*Integer*	patch boundary condition attribute pointer
KTYPE	*Integer*	patch type
KX	*Integer array*	location of patch extreme
PATRIB	*Real array*	patch attribute table

Inputs Through Common

Common Block /COMADP/

MBAINC	*Integer*	bc attribute table step
MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

D	*Real array*	D modified by boundary conditions
DDOT	*Real array*	DDOT modified by boundary conditions

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.2.2 Subroutine BCONE

External Routines

EXPERT
MESAGE

Intrinsic Functions

MAX
MOD

Arguments on Entry

E	*Real array*	electric field
KBLAS	*Integer array*	block addressing structure
KO	*Integer array*	location of patch origin
KTYPE	*Integer*	patch type
KX	*Integer array*	location of patch extreme

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

E	*Real array*	E modified by boundary conditions

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.2.3 Subroutine GLUEIO

External Routines

EXPERT
MESAGE

Intrinsic Functions

ISIGN
MAX
MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
KCASE	*Integer*	select copy/add to/from gluepatch
KO	*Integer array*	location of patch origin
KTYPE	*Integer*	patch type
KX	*Integer array*	location of patch extreme
PATCH	*Real array*	gluepatch buffer array
VEC	*Real array*	block vector field array

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

IC1	*Integer*	argument of MAX
ICLASS	*Integer*	argument of EXPERT
INGP	*Integer array*	argument of MAX
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

KLEN	*Integer*
PATCH	*Real array*
VEC	*Real array*

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.3 Particle routines

A.3.1 Subroutine SETCUR

External Routines

EXPERT	
NILVEC	set vector to zero

Intrinsic Functions

NONE

Arguments on Entry

KBLAS	*Integer array*	block addressing structure

Inputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT
KBLAS	*Integer*	argument of NILVEC
PCUR	*Real*	argument of NILVEC

Arguments on Exit

PCUR	*Real array*	initialised current array

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMDDP/

NONE

A.3.2 Subroutine MOVCUR

External Routines

ASSCUR	assign current to block element net
EXPERT	
IVAR	
MESAGE	
PBLINK	manage particle buffering between blocks

Intrinsic Functions

INT
MOD
SQRT

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
KCASE	*Integer*	0 for original move, > 0 for exchange buffer data
KCOVA	*Integer array*	block addressing for basis vectors
KPART	*Integer array*	particle addressing structure
PARTAT	*Real array*	particle attribute table
PBUFI	*Real array*	input particle buffer
PCOORD	*Real array*	particle coordinate array
PECOV	*Real array*	basis vector array

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MNMOM	*Integer*	loc of No of particle MOMentum coords in LPARAS
MNPOS	*Integer*	loc of No of particle POSition coords in LPARAS
MOCPS	*Integer*	offset for Charge Per Superparticle value
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MORGTI	*Integer*	Mesh ORIGin of 1/T in Ecov addressing
MPAINC	*Integer*	particle attribute table step
MPNOO	*Integer*	Particle NO Origin in LPARAS
MPORO	*Integer*	Particle Origin table Origin in LPARAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS

MSPEC	*Integer*	location of no of SPECies in LPARAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)
NOECA	*Integer*	no of ECOV addressing entries per component

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
IFACEK	*Integer*	argument of PBLINK
INC	*Integer array*	argument of ASSCUR
INXBUF	*Integer*	argument of PBLINK
IOFSET	*Integer*	argument of ASSCUR
IOPARO	*Integer*	argument of IVAR
IPOMAX	*Integer*	argument of IVAR
ISPACE	*Integer*	argument of ASSCUR
ISUB	*Integer*	argument of EXPERT
JPARTO	*Integer*	argument of IVAR
JSPEC	*Integer*	argument of PBLINK
NSTEP	*Integer*	argument of IVAR
PCUR	*Real*	argument of ASSCUR
ZCMULT	*Real*	argument of ASSCUR
ZXN	*Real array*	argument of ASSCUR
ZXO	*Real array*	argument of ASSCUR

Arguments on Exit

KPART	*Integer array*	particle addressing structure
PBUFO	*Real array*	output particle buffer
PCOORD	*Real array*	coordinate array
PCUR	*Real array*	current array

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NONE

Common Block /XXX/

NONE

Subroutine ASSCUR

External Routines

JOIN	GHOST routine (for testing)
LINCOL	GHOST routine (for testing) trajectory element
MERGE	merge ordered list of intersections
PICNOW	GHOST routine (for testing)
POSITN	GHOST routine (for testing)

Intrinsic Functions

INTS
MOD

Arguments on Entry

KINC	*Integer array*	block mesh increment
KOFSET	*Integer*	block mesh offset
KSPACE	*Integer*	block mesh space per component
PCMULT	*Real*	current/particle multiplier
PCUR	*Real array*	block current array
PXN	*Real array*	end position of virtual particle
PXO	*Real array*	start position of virtual particle

Inputs Through Common

Common Block /COMIBC/

NODIM	*Integer*	dimensionality
NOEL1	*Integer array*	no of elements in block type/side
XLEN1	*Real array*	length of side of block type

Common Block /COMGMA/

XYZBLK	*Real array*	global reference coordinate of block

Common Block /COMADP/

MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /XXX/

IBLOCK	*Integer*	current block number (for testing)

Arguments of Called Routines

ILEN	*Integer*	argument of MERGE
ILEN23	*Integer*	argument of MERGE
INX	*Integer array*	argument of MERGE
IOR1	*Integer*	argument of MERGE
IOR2	*Integer*	argument of MERGE
IOR3	*Integer*	argument of MERGE
IOR4	*Integer*	argument of MERGE
ZALFA	*Real array*	argument of MERGE
ZXN	*Real*	argument of JOIN
ZXO	*Real*	argument of POSITN
ZYN	*Real*	argument of JOIN
ZYO	*Real*	argument of POSITN

Arguments on Exit

PCUR	*Real array*	nodal contravariant currents

Outputs Through Common

Common Block /COMIBC/

NONE

Common Block /COMGMA/

NONE

Common Block /COMADP/

NONE

Common Block /XXX/

NONE

Subroutine MERGE

External Routines

NONE

Intrinsic Functions

NONE

Arguments on Entry

KA	*Integer*	length of list a
KB	*Integer*	length of list b
PA	*Real array*	list of increasing numbers a
PB	*Real array*	list of increasing numbers b

Inputs Through Common

NONE

Arguments of Called Routines

NONE

Arguments on Exit

KC	*Integer*	length of merged list c
PC	*Real*	ordered merged lists a and b

Outputs Through Common

NONE

A.3.3 Subroutine ACCEL

External Routines

EXPERT

Intrinsic Functions

MOD
SQRT

Arguments on Entry

B	*Real array*	scaled contravariant magnetic field
E	*Real array*	covariant electric field
KBLAS	*Integer array*	block addressing structure
KCOVA	*Integer array*	block addressing for basis vectors
KPART	*Integer array*	particle addressing structure
PARTAT	*Real array*	particle attribute table
PCOORD	*Real array*	particle coordinate array
PECOV	*Real array*	basis vector array

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MNMOM	*Integer*	loc of No of particle MOMentum coords in LPARAS
MNPOS	*Integer*	loc of No of particle POSition coords in LPARAS
MOCPS	*Integer*	offset for Charge Per Superparticle value
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MORGTI	*Integer*	Mesh ORIGin of 1/T in Ecov addressing
MORIGT	*Integer*	Mesh ORIGin of T in Ecov addressing
MPAINC	*Integer*	particle attribute table step
MPNOO	*Integer*	Particle NO Origin in LPARAS
MPORO	*Integer*	Particle Origin table Origin in LPARAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MSPEC	*Integer*	location of no of SPECies in LPARAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)
NOECA	*Integer*	no of ECOV addressing entries per component

NOECC	*Integer*	no of ECOV components per block

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PCOORD	*Real array*	particle coordinate array

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NONE

A.4 Particle boundary condition routines

A.4.1 Subroutine EMITEL

External Routines

EXPERT
MESAGE

Intrinsic Functions

MAX
MIN
MOD

Arguments on Entry

D	*Real array*	scaled contravariant displacement field
KBLAS	*Integer array*	block addressing structure
KO	*Integer array*	location of patch origin
KPART	*Integer array*	particle addressing structure
KX	*Integer array*	location of patch extreme
PCHG	*Real array*	charge density
PCPP	*Real*	charge per superparticle

Inputs Through Common

Common Block COMADP //

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MNMOM	*Integer*	loc of No of particle MOMentum coords in LPARAS
MNPOS	*Integer*	loc of No of particle POSition coords in LPARAS
MODKEY	*Integer*	Mesh orthogonality and dimension key location
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MSPEC	*Integer*	location of no of SPECies in LPARAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMBAS/

NONE

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

KLENO	*Integer*	length of PBUFO
PBUFO	*Real array*	gluepatch buffer containing new particles
PCHG	*Real array*	modified PCHG at cathode surface

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMBAS/

NONE

Common Block /COMDDP/

NONE

A.4.2 Subroutine QSHARE

External Routines

EXPERT

Intrinsic Functions

MOD

Arguments on Entry

KBLAS	*Integer array*	block addressing structure
KPART	*Integer array*	particle addressing structure
PARTAT	*Real array*	particle attribute table
PCHG	*Real array*	charge density
PCOORD	*Real array*	particle coordinate array

Inputs Through Common

Common Block /COMADP/

MINCO	*Integer*	Mesh INCrement Origin in BLAS
MNMOM	*Integer*	loc of No of particle MOMentum coords in LPARAS
MNPOS	*Integer*	loc of No of particle POSition coords in LPARAS
MOCPS	*Integer*	offset for Charge Per Superparticle value
MOFSET	*Integer*	Mesh OFfSET location in BLAS
MPAINC	*Integer*	particle attribute table step
MPNOO	*Integer*	Particle NO Origin in LPARAS
MPORO	*Integer*	Particle Origin table Origin in LPARAS
MSIZO	*Integer*	Mesh SIZe Origin in BLAS
MSPACE	*Integer*	Mesh SPACE reserved location in BLAS
MSPEC	*Integer*	location of no of SPECies in LPARAS
MXPDIM	*Integer*	Max number of physical dimensions(=3)

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NLOMT2	*Logical array*

Arguments of Called Routines

ICLASS	*Integer*	argument of EXPERT
ISUB	*Integer*	argument of EXPERT

Arguments on Exit

PCHG	*Real array*	charge density

Outputs Through Common

Common Block /COMADP/

NONE

Common Block /COMBAS/

NONE

Common Block /CHABAS/

NONE

Common Block /COMDDP/

NONE

B Annex 1: Report RFFX(92)52

RFFX(92)52

GENERAL GEOMETRY PIC FOR MIMD COMPUTERS

James W EASTWOOD
Roger W HOCKNEY
Wayne ARTER

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB
England

August 1992

Document Control Number: AEA/TLNA/31858/RP/1			
Date of Issue: June 21, 1993			Issue number: 2
Authorization	*Name*	*Signature*	*Position*
Prepared by	J.W. Eastwood		Project Manager
Approved by	D.E.T.F Ashby		Department Head

Contents

ABSTRACT

A new relativistic electromagnetic PIC algorithm for general two and three dimensional geometries is described. Correct choice of co- and contravariant field components and of weighting yields simple coordinate invariant numerical prescriptions. The combination of isoparametric hexahedral elements, generated by transfinite interpolation, and multiblock decomposition leads to algorithms ideally suited to MIMD computers.

Chapter 1

Introduction

1.1 Contents

Virtual Particle (VP) particle-mesh algorithms are now established as an effective approach to obtaining numerical schemes for solving the relativistic Maxwell-Vlasov equations [6, 7, 8, 2, 1, 9]. Unlike conventional Particle-in-Cell (PIC) schemes, they are derived using finite elements in both space and time. Current is assigned from 'virtual particles' placed at points specially interpolated between positions at successive time levels, a procedure which automatically leads to charge conservation. Existing VP implementations use rectangular finite elements in two dimensional cartesian and polar geometries. Only a restricted class of device is well modelled in such circumstances, leading to the need to implement VP in more complex geometries. This report shows how the VP algorithms extend to general three dimensional body-fitted elements [1]. The resulting multiblock decomposition of both field and particle data allow efficient usage of Distributed Memory MIMD architecture computers.

The report has been broken down into a number of largely self contained chapters, the contents of which are as follows:

Chapter 2 contains definitions and identities in general curvilinear coordinates used in the derivation of the equations for the isoparametric finite element particle-mesh schemes.

Chapter 3 describes the method of dividing complex objects into a set of curvilinear hexahedral blocks, and of subdividing those blocks into hexahedral finite elements using transfinite interpolation. Metric tensors and basis vectors are defined by using coordinate transformations isometric to the electric potential representation.

Chapter 4 outlines a number of alternative Virtual Particle schemes for the solution of the electromagnetic field equations and summarises the time step loop for the chosen variant.

Chapter 5 presents the Virtual Particle charge and current assignment schemes in general curvilinear coordinates for the linear basis function finite elements discussed in Chapter 4 for Maxwell's equations.

Chapter 6 describes the treatment of boundary conditions.

Chapter 7 outlines the planned data organisation in the MIMD implementation of the algorithm.

1.2 Acknowledgements

This report was prepared as part of a research project sponsored by the Air Force Office of Scientific Research (AFSC) under Contract F49620-92-C-0035, "General Geometry PIC Algorithms and Software for Distributed Memory MIMD Computers". A substantial part of the material contained in this report was developed from research undertaken during earlier research projects ABR 40 451 "Electromagnetic Modelling" sponsored by AEA Corporate Research and RAE 1B/7 "3-D Modelling of HPM Sources" sponsored by RAE Farnborough.

Chapter 2

Tensor Definitions

This Chapter contains definitions and identities in general curvilinear coordinates used in the derivation of the equations for the isoparametric finite element particle-mesh schemes.

2.1 Basis Vectors and Components

Let the reference cartesian coordinates have components $(x^1,\ x^2,\ x^3)$ and unit vectors $(\hat{\mathbf{x}}_1,\ \hat{\mathbf{x}}_2,\ \hat{\mathbf{x}}_3)$, so that a vector, $\mathbf{x}$, may be written

$$\mathbf{x} = x^i \hat{\mathbf{x}}_i \tag{2.1.1}$$

where summation over the repeated index i $(= 1, 2, 3)$ is implied. Now suppose that $(x^1,\ x^2,\ x^3)$ are expressed as functions of the curvilinear coordinate components $(\bar{x}^1,\ \bar{x}^2,\ \bar{x}^3)$, i.e.

$$x^1 = x^1\,(\bar{x}^1,\ \bar{x}^2,\ \bar{x}^3), \quad \text{etc}, \tag{2.1.2}$$

or more concisely

$$\mathbf{x} = \mathbf{x}(\bar{\mathbf{x}}) \tag{2.1.3}$$

We define the *basis vectors*

$$\mathbf{e}_i = \frac{\partial \mathbf{x}}{\partial \bar{x}^i} \tag{2.1.4}$$

and the *reciprocal basis vectors*

$$\mathbf{e}^i = \nabla \bar{x}^i \tag{2.1.5}$$

A vector $\mathbf{A}$ may be expressed in terms of its *contravariant* component A^i or *covariant* components A_i:-

$$\mathbf{A} = A^i \mathbf{e}_i : \text{contravariant} \tag{2.1.6}$$

$$= A_i \mathbf{e}^i : \text{covariant} \tag{2.1.7}$$

Note that in general $\mathbf{e}_i$ and $\mathbf{e}^i$ are *not* unit vectors. If we introduce the unit vectors.

$$\hat{\mathbf{e}}_i = \frac{\mathbf{e}_i}{|\mathbf{e}_i|} \tag{2.1.8}$$

then (2.1.6) may be written

$$\mathbf{A} = (A^i|\mathbf{e}_i|)\hat{\mathbf{e}}_i = A(i)\hat{\mathbf{e}}_i \tag{2.1.9}$$

where $A(i)$ are the *physical* components.

The basis vectors and reciprocal basis vectors are orthogonal

$$\mathbf{e}_i \cdot \mathbf{e}^j = \delta_i^j \tag{2.1.10}$$

where δ_j^i is the Kronecker delta (= 1 if $i = j$, 0 otherwise).

The reciprocal basis vectors can be written in terms of the basis vectors

$$\mathbf{e}^i = \frac{\mathbf{e}_j \times \mathbf{e}_k}{|\mathbf{e}_i \cdot \mathbf{e}_j \times \mathbf{e}_k|} = \frac{\mathbf{e}_j \times \mathbf{e}_k}{J} \tag{2.1.11}$$

and vice-versa

$$\mathbf{e}_i = (\mathbf{e}^j \times \mathbf{e}^k)J \tag{2.1.12}$$

where the Jacobian

$$J = \sqrt{g} = |\mathbf{e}_i \cdot \mathbf{e}_j \times \mathbf{e}_k| \tag{2.1.13}$$

can be written in terms of the square root of the determinant, g, of the metric tensor.

The (covariant) metric tensor is defined as

$$g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j \tag{2.1.14}$$

and its reciprocal tensor, the contravariant metric tensor is

$$g^{ij} = \mathbf{e}^i \cdot \mathbf{e}^j \tag{2.1.15}$$

It follows that

$$g_{ik}g^{kj} = \delta_i^j \tag{2.1.16}$$

$$g = ||g_{ij}|| = J^2 \tag{2.1.17}$$

$$A^i = g^{ij}A_j \tag{2.1.18}$$

$$A_i = g_{ij}A^j \tag{2.1.19}$$

In cartesian coordinates, the covariant, contravariant and physical components are identical and the metric tensor reduces to the $g_{ij} = \delta_{ij}$. For orthogonal systems, $g_{ij} = 0$ for $i \neq j$. In general g_{ij} is symmetric with six distinct elements, $g_{ij} = g_{ji}$.

2.2 Coordinate Transformation

A vector **A** is independent of the coordinate system it is represented in. Thus if **A** has contravariant components A^i, coordinates x^i, and bases $\mathbf{e}_i$ in one system, and $\bar{A}^i$, $\bar{x}^i$, $\bar{\mathbf{e}}_i$ in another, then

$$\mathbf{A} = A^i\mathbf{e}_i = \bar{A}^i\bar{\mathbf{e}}_i \tag{2.2.1}$$

Dotting with $\mathbf{e}^j$ gives

$$A^j = \bar{A}^i\bar{\mathbf{e}}_i \cdot \mathbf{e}^j \tag{2.2.2}$$

i.e.

$$A^j = \bar{A}^i\frac{\partial x^j}{\partial \bar{x}^i} \tag{2.2.3}$$

Similarly

$$A_j = \bar{A}_i\frac{\partial \bar{x}^i}{\partial x^j} \tag{2.2.4}$$

Tensors transform similarly. So for example

$$B_{ij} = \frac{\partial \bar{x}^k}{\partial x^i}\frac{\partial \bar{x}^l}{\partial x^j}\bar{B}_{kl} \tag{2.2.5}$$

where B_{ij} and $\bar{B}_{kl}$ are rank 2 tensors in the unbarred and barred coordinate systems, respectively.

From Eq.(2.2.5), it follows that if we have $\mathbf{x} = \mathbf{x}(\bar{\mathbf{x}})$, where $\mathbf{x}$ are cartesian, and given that in cartesians, $g_{ij} = \delta_{ij}$, then in the barred coordinates the metric tensor is

$$\bar{g}_{kl} = \frac{\partial x^i}{\partial \bar{x}^k}\frac{\partial x^j}{\partial \bar{x}^l}\delta_{ij} = \frac{\partial x^i}{\partial \bar{x}^k}\frac{\partial x^i}{\partial \bar{x}^l} \tag{2.2.6}$$

which is the same as given by Eq.(2.1.14).

Volume elements transform in the usual fashion:

$$d\tau = dx^1\, dx^2\, dx^3 \;=\; J\, d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3 \tag{2.2.7}$$

where

$$J = \sqrt{g} = \left|\frac{\partial \mathbf{x}}{\partial \bar{\mathbf{x}}}\right|$$

2.3 Vector Identities

$a^i, \mathbf{e}_i$: contravariant components and bases

$a_i, \mathbf{e}^i$: covariant components and bases

g_{ij} : metric tensor, $g = ||g_{ij}||$

ρ : scalar

$$\left.\begin{matrix} e^{ijk} \\ e_{ijk} \end{matrix}\right\} \text{permutation symbols: } = 1 \text{ for cyclic indices, -1 for anticyclic and 0 for repeated indices} \tag{2.3.1}$$

$$\epsilon^{ijk} : \text{permutation tensor} = e^{ijk}/\sqrt{g} \tag{2.3.2}$$

$$\epsilon_{ijk} : \text{permutation tensor} = \sqrt{g} e_{ijk} \tag{2.3.3}$$

$$\mathbf{a}\cdot\mathbf{b} = a^i b_i = a_i b^i \tag{2.3.4}$$

$$(\mathbf{a}\times\mathbf{b})^i = \epsilon^{ijk} a_j b_k \tag{2.3.5}$$

$$(\mathbf{a}\times\mathbf{b})_i = \epsilon_{ijk} a^j b^k \tag{2.3.6}$$

$$(\nabla\rho)_i = \frac{\partial\rho}{\partial\bar{x}^i} \tag{2.3.7}$$

$$\nabla\cdot\mathbf{a} = \frac{1}{\sqrt{g}}\frac{\partial}{\partial\bar{x}^i}\sqrt{g}a^i \tag{2.3.8}$$

$$\nabla\cdot\left(\frac{\mathbf{e}_i}{\sqrt{g}}\right) \equiv 0 \tag{2.3.9}$$

$$(\nabla\times\mathbf{a})^i = \epsilon^{ijk}\frac{\partial a_k}{\partial\bar{x}^j} \tag{2.3.10}$$

$$\mathbf{a}\cdot\nabla\rho = a^i\frac{\partial}{\partial\bar{x}^i}\rho \tag{2.3.11}$$

$$\nabla^2\rho = \frac{1}{\sqrt{g}}\frac{\partial}{\partial\bar{x}^i}\left(g^{ij}\sqrt{g}\frac{\partial\rho}{\partial\bar{x}^j}\right) \tag{2.3.12}$$

$$\epsilon^{ijk}\epsilon_{klm} = \delta^i_l\delta^j_m - \delta^i_m\delta^j_l \tag{2.3.13}$$

2.4 Derivatives of Vectors

Given some vector **A**, then its derivative with respect to some cartesian component x^i is

$$A^i_{,j} = \hat{\mathbf{x}}^i\cdot\frac{\partial}{\partial x^j}\mathbf{A} = \frac{\partial A^i}{\partial x^j} \tag{2.4.1}$$

In general (barred) coordinates, the covariant derivative is likewise defined,

$$A^i_{;j} = \mathbf{e}^i\cdot\frac{\partial}{\partial\bar{x}^j}\mathbf{A} = \mathbf{e}^i\cdot\frac{\partial}{\partial\bar{x}^j}\mathbf{e}_k A^k \tag{2.4.2}$$

but now the basis vectors are no longer independent of the coordinates. From (2.4.2)

$$A^i_{;j} = \frac{\partial A^i}{\partial\bar{x}^j} + A^k\left\{\mathbf{e}^i\cdot\frac{\partial}{\partial\bar{x}^j}\mathbf{e}_k\right\} \tag{2.4.3}$$

The term is the curly brackets in Eq.(2.4.3) is the *Christoffel symbol of the first kind*. From Eqs.(2.1.4) and (2.1.5) we have

$$\mathbf{e}^i\cdot\frac{\partial\mathbf{e}_k}{\partial\bar{x}^j} = \frac{\partial\bar{x}^i}{\partial x^l}\frac{\partial^2 x^l}{\partial\bar{x}^j\partial\bar{x}^k} = \Gamma^i_{jk} \tag{2.4.4}$$

The Christoffel symbols are *not* tensors, and they transform between two general coordinate systems according to

$$\bar{\Gamma}^i_{jk} = \Gamma^s_{pq}\frac{\partial \bar{x}^i}{\partial x^s}\frac{\partial x^p}{\partial \bar{x}^j}\frac{\partial x^q}{\partial \bar{x}^k} + \frac{\partial \bar{x}^i}{\partial x^l}\frac{\partial^2 x^l}{\partial \bar{x}^j \partial \bar{x}^k} \tag{2.4.5}$$

Repeating the above argument for covariant vector components we have

$$A_{i,j} = \frac{\partial A_i}{\partial \bar{x}^j} + A_k \mathbf{e}_i \cdot \frac{\partial \mathbf{e}^k}{\partial \bar{x}^j} \tag{2.4.6}$$

$$= \frac{\partial A_i}{\partial \bar{x}^j} - A_k \mathbf{e}^k \cdot \frac{\partial \mathbf{e}_i}{\partial \bar{x}^j} \tag{2.4.7}$$

$$= \frac{\partial A_i}{\partial \bar{x}^j} - \Gamma^k_{ij} A_k \tag{2.4.8}$$

To obtain (2.4.7), we used $\mathbf{e}^i \cdot \mathbf{e}_i = \delta^k_i$ and differentiated term by term.

The Christoffel symbols arise when we wish to write terms such as $\mathbf{a} \cdot \nabla \mathbf{b}$ in tensor form, or in the absolute (or intrinsic) time derivative of a vector.

The covariant components of the absolute time derivative of vector $\mathbf{A}$ are defined as

$$\frac{\delta A_p}{\delta t} = \mathbf{e}_p \cdot \frac{d\mathbf{A}}{dt} = \mathbf{e}_p \cdot \frac{d}{dt}\mathbf{e}^r A_r$$

$$= \frac{dA_p}{dt} + A_r \left\{ \mathbf{e}_p \cdot \frac{\partial \mathbf{e}^r}{\partial \bar{x}^q} \right\} \frac{d\bar{x}^q}{dt}$$

$$= \frac{dA_p}{dt} - \Gamma^r_{pq} A_r \frac{d\bar{x}^q}{dt} \tag{2.4.9}$$

The contravariant components are likewise defined

$$\frac{\delta A^p}{\delta t} = \mathbf{e}^p \cdot \frac{d\mathbf{A}}{dt} = \mathbf{e}^p \cdot \frac{d}{dt}\mathbf{e}_r A^r$$

$$= \frac{dA^p}{dt} + A^r \left(\mathbf{e}^p \cdot \frac{d\mathbf{e}_r}{dt} \right)$$

$$= \frac{dA^p}{dt} + A^r \left\{ \mathbf{e}^p \cdot \frac{\partial \mathbf{e}_r}{\partial \bar{x}^q} \right\} \frac{d\bar{x}^q}{dt}$$

$$= \frac{dA^p}{dt} + \Gamma^p_{qr} A^r \frac{d\bar{x}^q}{dt} \tag{2.4.10}$$

2.5 Maxwell's Equations

Using the formulae of Section 2.3 we can write Maxwell's equations in tensor form:-

$$\frac{\partial B^i}{\partial t} = -\epsilon^{ijk}\frac{\partial E_k}{\partial \bar{x}^j} \tag{2.5.1}$$

$$\frac{1}{\sqrt{g}}\frac{\partial}{\partial \bar{x}^i}\sqrt{g}B^i = 0 \tag{2.5.2}$$

$$\frac{\partial D^i}{\partial t} = \epsilon^{ijk}\frac{\partial H_k}{\partial \bar{x}^j} - j^i \tag{2.5.3}$$

$$\frac{1}{\sqrt{g}}\frac{\partial}{\partial \bar{x}^i}\sqrt{g}D^i = \rho \tag{2.5.4}$$

where B^i and E_k are magnetic and electric field components, D^i are electric displacement components and H_k are magnetic intensity components. Quantities j^i and ρ are respectively current density and charge density.

It is convenient to introduce extensive current and charge variables

$$I^i = \sqrt{g}j^i \tag{2.5.5}$$

$$Q = \sqrt{g}\rho \tag{2.5.6}$$

and volume scaled flux quantities b^i and d^i for magnetic and displacement fields

$$b^i = \sqrt{g}B^i \tag{2.5.7}$$

$$d^i = \sqrt{g}D^i \tag{2.5.8}$$

If in addition we write the permutation tensor in terms of the permutation symbol, we can write Maxwell's equations as

$$\frac{\partial b^i}{\partial t} = -e^{ijk}\frac{\partial E_k}{\partial \bar{x}^j} \tag{2.5.9}$$

$$\frac{\partial b^i}{\partial \bar{x}^i} = 0 \tag{2.5.10}$$

$$\frac{\partial d^i}{\partial t} = e^{ijk}\frac{\partial H_k}{\partial \bar{x}^j} - I^i \tag{2.5.11}$$

$$\frac{\partial d^i}{\partial \bar{x}^i} = Q \tag{2.5.12}$$

Equations (2.5.9) - (2.5.12) have the particularly attractive feature that they have the same form irrespective of coordinate system, and contain no explicit reference to the metrics. This is also true of the relationships between the electromagnetic fields and potentials:-

$$E_k = -\frac{\partial \phi}{\partial \bar{x}^k} - \dot{A}_k \tag{2.5.13}$$

$$b^i = -e^{ijk}\frac{\partial A_k}{\partial \bar{x}^j} \tag{2.5.14}$$

The only explicit reference to the metrics appears in the constitutive relationships between fields, which in vacuo are

$$\mu_0 H_k = \frac{g_{k\ell}}{\sqrt{g}}b^\ell \tag{2.5.15}$$

$$\epsilon_0 E_k = \frac{g_{k\ell}}{\sqrt{g}} d^\ell \tag{2.5.16}$$

More generally, μ_0 and ϵ_0 can be replaced by tensor permeabilities and permittivities.

A similar attractive simplicity appears in the Action Integral for the electromagnetic field equations:-

$$I = \int dt\, d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3 \left\{ \frac{1}{2}(E_i d^i - H_i b^i) + I^i A_i - Q\phi \right\} \tag{2.5.17}$$

The metric free forms simplifies the problem of writing a program module for solving Maxwell's equations in arbitrary non-orthogonal coordinate systems.

2.6 Equations of Motion

The relativistic equations of motion of a charged particle are

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{2.6.1}$$

$$\frac{d\mathbf{p}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \tag{2.6.2}$$

where

$$\mathbf{p} = \gamma m_0 \mathbf{v} \tag{2.6.3}$$

$$\gamma^2 = 1 + \frac{|p|^2}{(m_0 c)^2} = 1 / \left(1 - \frac{|v|^2}{c^2}\right) \tag{2.6.4}$$

Introducing general coordinates, $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$, i.e.

$$\mathbf{x} = \mathbf{x}(\bar{x}^1, \bar{x}^2, \bar{x}^3) \tag{2.6.5}$$

gives

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \frac{\partial \mathbf{x}}{\partial \bar{x}^k} \frac{d\bar{x}^k}{dt} \\ &= \mathbf{e}_k \frac{d\bar{x}^k}{dt} = \mathbf{e}_k \bar{v}^k \end{aligned} \tag{2.6.6}$$

Thus, Eq.(2.6.1) transforms to

$$\frac{d\bar{x}^k}{dt} = \bar{v}^k \tag{2.6.7}$$

Applying the results of Eqs.(2.3.3), (2.3.6), (2.4.9) and (2.5.7) to Eq.(2.6.2) gives the covariant momentum equation

$$\frac{d\bar{p}_s}{dt} - \Gamma^r_{sq} \bar{p}_r \bar{v}^q = q\left[E_s + e_{sqr} \bar{v}^q b^r\right] \tag{2.6.8}$$

The particle dynamics can be included in the action integral, Eq.(2.5.17), by adding the extra particle lagrangian term,

$$-\int \frac{m_0 c^2}{\gamma} dt \tag{2.6.9}$$

and explicitly expressing the charges and currents in Eq.(2.5.17) as sums over particles, i, with charges q_i

$$Q = \sum_i q_i \delta(\bar{x}_i^1 - \bar{x}^1)\delta(\bar{x}_i^2 - \bar{x}^2)\delta(\bar{x}_i^3 - \bar{x}^3) \tag{2.6.10}$$

$$I^m = \sum_i q_i \delta(\bar{x}_i^1 - \bar{x}^1)\delta(\bar{x}_i^2 - \bar{x}^2)\delta(\bar{x}_i^3 - \bar{x}^3)\frac{d\bar{x}^m}{dt} \tag{2.6.11}$$

The equation of motion, Eq.(2.6.8) arises from the Euler-Lagrange equation

$$\frac{\partial}{\partial \bar{x}^s} L - \frac{d}{dt}\frac{\partial L}{\partial \dot{\bar{x}}^s} = 0 \tag{2.6.12}$$

where L is the total Lagrangian density.

Chapter 3

Finite Element Generation

This Chapter describes the method of dividing complex objects into a set of curvilinear hexahedral blocks, and of subdividing those blocks into hexahedral finite elements using transfinite interpolation. Metric tensors and basis vectors are defined by using coordinate transformations isometric to the electric potential representation.

3.1 Introduction

A two level decomposition of a complex object into a set of finite elements is proposed. First, the object is divided into a set of curvilinear hexahedral blocks. In selecting this *multiblock* division, it is important to choose the division of complex objects into sufficiently convex volumes to avoid the creation of very small, or even negative, volume elements in the blending process; this could cause the numerical simulations to fail. However in the applications envisaged it is unlikely that singular elements will present serious difficulties. Computational efficiency dictates that wherever possible, orthogonal blocks be used (such as rectangular bricks and cylinder sections) as this greatly reduces computational costs and storage. For the purposes of facilitating load balancing on MIMD computers, it may be advantageous to subdivide blocks beyond the level dictated by the object's geometry.

Each block of the multiblock decomposition is described by its bounding curves; four intersecting curves in two dimensions and twelve curves in three dimensions. Alternatively, the three dimensional block may be described by bounding surfaces. The relationship of the block coordinates to the global coordinates of the object is described in Section 3.5. The locations of element nodes within a block are generated by blending the interpolants from the bounding curves or surfaces, as described in Sections 3.3 and 3.4.

In order to transform between curved space (barred) and physical coordinates, and to integrate the equations of motion, values of the contravariant basis vectors (Eq.(2.1.4)) are required. The integration of the field equations requires metric tensor values (Eq.(2.1.4)). These may be evaluated from the finite element approximations to the coordinate transformations, as outlined in the following section.

3.2 Isoparametric Elements

3.2.1 2-D: Linear Quadrilaterals

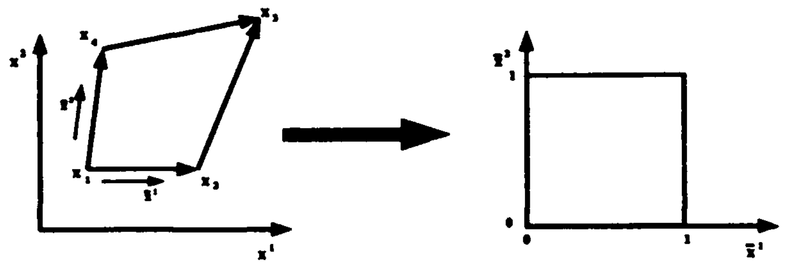


Figure 3.1: *The isoparametric linear quadrilateral element with corner nodes* $\mathbf{x}_1 \ldots \mathbf{x}_4$ *maps to the unit square in the barred coordinate space.*

The two dimensional linear isoparametric element uses the same linear support function for the coordinate mapping as for the finite element scalar potential. Figure 3.1 shows a quadrilateral element in physical and barred coordinate space. A point $\mathbf{x}$ is related to the barred coordinates $\bar{\mathbf{x}} = (\bar{x}^1, \bar{x}^2)$ by

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1(1-\bar{x}^1)(1-\bar{x}^2) \\ &+ \mathbf{x}_2\bar{x}^1(1-\bar{x}^2) \\ &+ \mathbf{x}_3\bar{x}^1\bar{x}^2 \\ &+ \mathbf{x}_4(1-\bar{x}^1)\bar{x}^2 \end{aligned} \tag{3.2.1}$$

Contravariant bases

$$\begin{aligned} \mathbf{e}_1 = \frac{\partial \mathbf{x}}{\partial \bar{x}^1} &= (\mathbf{x}_2-\mathbf{x}_1)(1-\bar{x}^2) + (\mathbf{x}_3-\mathbf{x}_4)\bar{x}^2 \\ &= \mathbf{r}_S(1-\bar{x}^2) + \mathbf{r}_N\bar{x}^2 \end{aligned} \tag{3.2.2}$$

$$\mathbf{e}_2 = \frac{\partial \mathbf{x}}{\partial \bar{x}^2} = \mathbf{r}_W(1-\bar{x}^1) + \mathbf{r}_E\bar{x}^1 \tag{3.2.3}$$

$$\mathbf{e}_3 = \hat{\mathbf{z}} \tag{3.2.4}$$

Covariant bases

$$\mathbf{e}^1 = \left(\frac{\mathbf{e}_2 \times \hat{z}}{J}\right) = \left[(1-\bar{x}^1)\mathbf{r}_W \times \hat{\mathbf{z}} + \bar{x}^1\mathbf{r}_E \times \hat{\mathbf{z}}\right]/J \tag{3.2.5}$$

$$\mathbf{e}^2 = \left(\frac{\hat{\mathbf{z}} \times \mathbf{e}_1}{J}\right) = \left[(1-\bar{x}^2)\hat{\mathbf{z}} \times \mathbf{r}_S + \bar{x}^2 \hat{\mathbf{z}} \times \mathbf{r}_N\right]/J \tag{3.2.6}$$

$$\mathbf{e}^3 = \hat{\mathbf{z}} \tag{3.2.7}$$

Metric tensor

$$g_{11} = \mathbf{e}_1 \cdot \mathbf{e}_1 = (1-\bar{x}^2)^2 r_S^2 + (\bar{x}^2)^2 r_N^2 + 2\bar{x}^2(1-\bar{x}^2)\mathbf{r}_S \cdot \mathbf{r}_N \tag{3.2.8}$$

$$g_{22} = \mathbf{e}_2 \cdot \mathbf{e}_2 = (1-\bar{x}^1)^2 r_W^2 + (\bar{x}^1)^2 r_E^2 + 2\bar{x}^1(1-\bar{x}^1)\mathbf{r}_E \cdot \mathbf{r}_W \tag{3.2.9}$$

$$g_{33} = 1 \tag{3.2.10}$$

$$\begin{aligned} g_{12} = g_{21} &= \mathbf{e}_2 \cdot \mathbf{e}_1 \\ = (1-\bar{x}^1)(1-\bar{x}^2)\mathbf{r}_S \cdot \mathbf{r}_W &+ \bar{x}^1(1-\bar{x}^2)\mathbf{r}_S \cdot \mathbf{r}_E + \bar{x}^1\bar{x}^2\mathbf{r}_N \cdot \mathbf{r}_E + (1-\bar{x}^1)\bar{x}^2\mathbf{r}_N \cdot \mathbf{r}_W \end{aligned} \tag{3.2.11}$$

$$g_{23} = g_{32} = g_{13} = g_{31} = 0 \tag{3.2.12}$$

$$\begin{aligned} J = \sqrt{g} &= \mathbf{e}_1 \times \mathbf{e}_2 \cdot \mathbf{e}_3 \\ &= (1-\bar{x}^1)(1-\bar{x}^2)(\mathbf{r}_S \times \mathbf{r}_W) \cdot \hat{\mathbf{z}} \\ &+ \bar{x}^1(1-\bar{x}^2)(\mathbf{r}_S \times \mathbf{r}_E) \cdot \hat{\mathbf{z}} \\ &+ \bar{x}^1\bar{x}^2(\mathbf{r}_N \times \mathbf{r}_E) \cdot \hat{\mathbf{z}} \\ &+ (1-\bar{x}^1)\bar{x}^2(\mathbf{r}_N \times \mathbf{r}_W) \cdot \hat{\mathbf{z}} \end{aligned} \tag{3.2.13}$$

The formulae (3.2.2) - (3.2.13) give a geometrical interpretation of the basis vectors and metric elements in terms of interpolation in the element side vectors $\mathbf{r}_W$, $\mathbf{r}_S$, $\mathbf{r}_E$ and $\mathbf{r}_W$.

A visualisation of the reciprocal vector sets is provided by Figure 3.2. There, both the contravariant and covariant basis vectors are sketched for node 1 of the element. At node 1, the Jacobian takes a value equal to the area of the parallelogram formed by $\mathbf{r}_W$ and $\mathbf{r}_S$:

$$J = J_{00} = \mathbf{r}_S \times \mathbf{r}_W \cdot \hat{\mathbf{z}} \tag{3.2.14}$$

and the vectors are

$$\mathbf{e}^1 = \mathbf{r}_W \times \hat{\mathbf{z}}/J_{00}, \quad \mathbf{e}_1 = \mathbf{r}_S \tag{3.2.15}$$

$$\mathbf{e}^2 = \hat{\mathbf{z}} \times \mathbf{r}_S/J_{00}, \quad \mathbf{e}_2 = \mathbf{r}_W \tag{3.2.16}$$

3.2.2 3-D: Linear Hexahedra

If we label the nodes of the 3-D element by index triplet (i,j,k), $i,j,k = 0$ or 1, and let

$$W_0(z) = (1-z) \tag{3.2.17}$$

$$W_1(z) = z \tag{3.2.18}$$

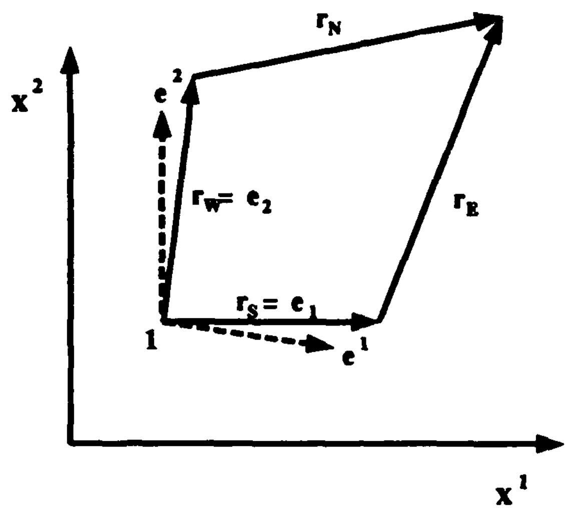


Figure 3.2: *A sketch of the basis vectors at node 1 of the quadrilateral element*

then the 3-D analogue of Eq.(3.2.1) becomes

$$\mathbf{x} = \mathbf{x}_{ijk} W_i(\bar{x}^1) W_j(\bar{x}^2) W_k(\bar{x}^3) \tag{3.2.19}$$

The contravariant basis vectors become

$$\begin{aligned} \mathbf{e}_1 = \frac{\partial \mathbf{x}}{\partial \bar{x}^1} &= (\mathbf{x}_{1jk} - \mathbf{x}_{0jk}) W_j(\bar{x}^2) W_k(\bar{x}^3) \\ &= \mathbf{r}_{\frac{1}{2}jk} W_j(\bar{x}^2) W_k(\bar{x}^3) \end{aligned} \tag{3.2.20}$$

Similarly

$$\mathbf{e}_2 = \mathbf{r}_{i\frac{1}{2}k} W_i(\bar{x}^1) W_k(\bar{x}^3) \tag{3.2.21}$$

$$\mathbf{e}_3 = \mathbf{r}_{ij\frac{1}{2}} W_i(\bar{x}^1) W_j(\bar{x}^2) \tag{3.2.22}$$

The covariant basis functions, metric tensor elements, etc follow by substituting Eq.(3.2.19) into the appropriate formula as shown in Section 4.1.

3.3 Two Dimensional Interpolation

Mesh generation for finite volume and finite element schemes, by blending the interpolants from intersecting pairs of curves such that the mesh exactly fits the boundary curves, was introduced by Gordon & Hall [1] and is now widely used in body fitting fluid flow codes, such as Harwell- FLOW3D [2,3]. We summarise here the transfinite interpolation formulae that we plan to use in the body fitting electromagnetic particle-mesh software.

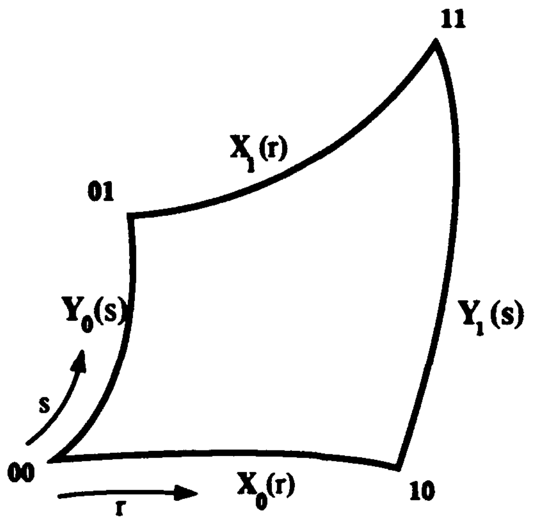


Figure 3.3: *The four-sided domain is bounded by two pairs, $(\mathbf{X}_0, \mathbf{X}_1)$ and $(\mathbf{Y}_0, \mathbf{Y}_1)$, of curves with intersections at the four points $\mathbf{x}_{ij}$, $i, j = 0$ or 1.*

Figure 3.3 shows a four sided domain. Each of the boundary curve pairs is a function of a single variable, denoted r or s. It is assumed, without loss of generality, that $0 \le r \le 1$, $0 \le s \le 1$. The problem is to define a function $\mathbf{x}(r,s)$ such that

$$\mathbf{x}(r,0) = \mathbf{X}_0(r) \qquad (3.3.1)$$

$$\mathbf{x}(0,s) = \mathbf{Y}_0(s) \qquad (3.3.2)$$

$$\mathbf{x}(r,1) = \mathbf{X}_1(r) \qquad (3.3.3)$$

$$\mathbf{x}(1,s) = \mathbf{Y}_1(s) \qquad (3.3.4)$$

and

$$\mathbf{x}(i,j) = \mathbf{X}_j(i) = \mathbf{Y}_i(j) = \mathbf{x}_{ij} \qquad (3.3.5)$$

where $i, j = 0$ or 1.

If we introduce interpolating functions $\psi_0(s)$ and $\psi_1(s)$, where $\psi_0(0) = 1$, $\psi_0(1) = 0$, $\psi_1(0) = 0$, $\psi_1(1) = 1$, then we can write the 'horizontal' curve interpolation

$$\mathbf{x}_h(r,s) = \mathbf{X}_0(r)\psi_0(s) + \mathbf{X}_1(r)\psi_1(s) \qquad (3.3.6)$$

(Typically, one would take ψ piecewise linear, i.e. $\psi_0 = 1 - s$, $\psi_1 = s$). Similarly, we can define the 'vertical' curve interpolation using interpolation functions $\phi_0(r)$ and $\phi_1(r)$:-

$$\mathbf{x}_v(r,s) = \phi_0(r)\mathbf{Y}_0(s) + \phi_1(r)\mathbf{Y}_1(s) \qquad (3.3.7)$$

The interpolation given by (3.3.6) satisfies (3.3.1) and (3.3.3), but not in general (3.3.2) and (3.3.4). Similarly, Eq.(3.3.7) satisfies (3.3.2) and (3.3.4), but not generally (3.3.1) and (3.3.3). Both interpolants (Eqs.(3.3.6) and (3.3.7)) satisfy the corner constraints, Eq.(3.3.5), as does the corner interpolant

$$\mathbf{x}_c(r,s) = \mathbf{x}_{00}\phi_0(r)\psi_0(s) + \mathbf{x}_{01}\phi_0(r)\psi_1(s) + \mathbf{x}_{10}\phi_1(r)\psi_0(s) + \mathbf{x}_{11}\phi_1(r)\psi_1(s) \tag{3.3.8}$$

Taking a linear combination of Eqs.(3.3.6)-(3.3.8) gives

$$\mathbf{x} = \alpha\mathbf{x}_h + \beta\mathbf{x}_v + \gamma\mathbf{x}_c \tag{3.3.9}$$

Evaluating Eq.(3.3.9) at the sides $r = 0$, 1, and $s = 0, 1$, and equating the results to Eqs.(3.3.1)-(3.3.5) yields the desired transfinite interpolation function if $\alpha = \beta = -\gamma = 1$. Thus we may write our transfinite interpolation function

$$\mathbf{x}(r,s) = \mathbf{X}_j(r)\psi_j(s) + \mathbf{Y}_i(s)\phi_i(r) - \mathbf{x}_{ij}\phi_i(r)\psi_j(s) \tag{3.3.10}$$

where the sums over $i, j = 0$ or 1 are implied.

3.3.1 A Tabular Curve Example

Given the 'horizontal' tabular curves

$$\mathbf{X}_0(I), \mathbf{X}_1(I), \quad I = 0, NX \tag{3.3.11}$$

and 'vertical' tabular curves

$$\mathbf{Y}_0(J),\ \mathbf{Y}_1(J),\ J = 0, NY \tag{3.3.12}$$

where

$$\begin{aligned} \mathbf{X}_0(0) = \mathbf{Y}_0(0), &\quad \mathbf{X}_1(0) = \mathbf{Y}_0(NY) \\ \mathbf{X}_0(NX) = \mathbf{Y}_1(0), &\quad \mathbf{X}_1(NX) = \mathbf{Y}_1(NY) \end{aligned} \tag{3.3.13}$$

and assuming linear interpolation, we shall obtain the expression for the coordinates of node (I,J).

From the definitions of r, s, I, J, we have

$$r = I/NX, \quad s = J/NY \tag{3.3.14}$$

For linear interpolation

$$\phi_0(z) = \psi_0(z) = 1 - z; \quad 0 \le z \le 1 \tag{3.3.15}$$

$$\phi_1(z) = \psi_1(z) = z; \quad 0 \le z \le 1 \tag{3.3.16}$$

$$\begin{aligned} \mathbf{x}(I,J) &= \mathbf{X}_0(I)(1-s) + \mathbf{X}_1(I)s \\ &+ \mathbf{Y}_0(J)(1-r) + \mathbf{Y}_1(J)r \\ &- \mathbf{x}_{00}(1-r)(1-s) - \mathbf{x}_{01}(1-r)s - \mathbf{x}_{10}r(1-s) - \mathbf{x}_{11}rs \end{aligned} \tag{3.3.17}$$

3.3.2 Coordinate Transformation

The transfinite interpolation example given above can be readily extended to provide global definitions of basis vectors and metric tensor elements in terms of the boundary data values. If we assume that the domain enclosed by the 'horizontal' and 'vertical' curve pairs maps to the rectangle $0 \le \bar{x}^1 \le NX$, $0 \le \bar{x}^2 \le NY$ in curved space (so the curved space grid would have unit square elements), and that points on the boundary curves are joined by straight segments, then

$$\mathbf{X}_0(\bar{x}^1) = \mathbf{X}_0(I) + (\mathbf{X}_0(I+1) - \mathbf{X}_0(I))(\bar{x}^1 - I) \qquad (3.3.18)$$

for $I < \bar{x}^1 \le I+1$, etc, and (17) may be extended to general position $(\bar{x}^1, \bar{x}^2)$:-

$$\begin{aligned} \mathbf{x}(\bar{x}^1,\bar{x}^2) &= \mathbf{X}_0(\bar{x}^1) + (\mathbf{X}_1(\bar{x}^1) - \mathbf{X}_0(\bar{x}^1) - \mathbf{x}_{01})\bar{x}^2/NY \\ &+ \mathbf{Y}_0(\bar{x}^2) + (\mathbf{Y}_1(\bar{x}^2) - \mathbf{Y}_0(\bar{x}^2) - \mathbf{x}_{10})\bar{x}^1/NX \\ &- \mathbf{x}_{00} + (\mathbf{x}_{01} + \mathbf{x}_{10} - \mathbf{x}_{00} - \mathbf{x}_{11})\bar{x}^1\bar{x}^2/NX\ NY \qquad (3.3.19) \end{aligned}$$

from which values of contravariant basis vectors

$$\mathbf{e}_i = \frac{\partial \mathbf{x}}{\partial \bar{x}^i} \qquad (3.3.20)$$

and metric tensor elements

$$g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j \qquad (3.3.21)$$

may be computed.

3.4 Three Dimensional Interpolation

3.4.1 Interpolation between Specified Curves

The two dimensional interpolation scheme of the previous section generalises straightforwardly to three dimensions. The aim is to divide the general curvilinear hexahedra as illustrated in Figure 3.4 into a set of hexahedral elements which map to unit cubes in curved space.

The bounding curve $\mathbf{X}_{ij}(r)$, $r\epsilon[0,1]$ joins the corner node $\mathbf{x}_{0ij}$ at $r = 0$ to node $\mathbf{x}_{1ij}$ at $r = 1$. Similarly $\mathbf{Y}_{ij}(s)$ joins $\mathbf{x}_{i0j}$ to $\mathbf{x}_{i1j}$ and $\mathbf{Z}_{ij}(t)$ joins $\mathbf{x}_{ij0}$ to $\mathbf{x}_{ij1}$, where $i,j = 0$ or 1. The interpolation functions for the r, s, and t coordinates are respectively ϕ, ψ and η.

Blending interpolants between the bounding curves to obtain exact fits at all eight edges gives the formula

$$\begin{aligned} \mathbf{x}(r,s,t) &= \mathbf{X}_{jk}(r)\psi_j(s)\eta_k(t) + \mathbf{Y}_{ik}(s)\phi_i(r)\eta_k(t) \\ &+ \mathbf{Z}_{ij}(t)\phi_i(r)\psi_j(s) - 2\mathbf{x}_{ijk}\phi_i(r)\psi_j(s)\eta_k(t) \qquad (3.4.1) \end{aligned}$$

where sums over $i,j,k = 0$ and 1 are implied.

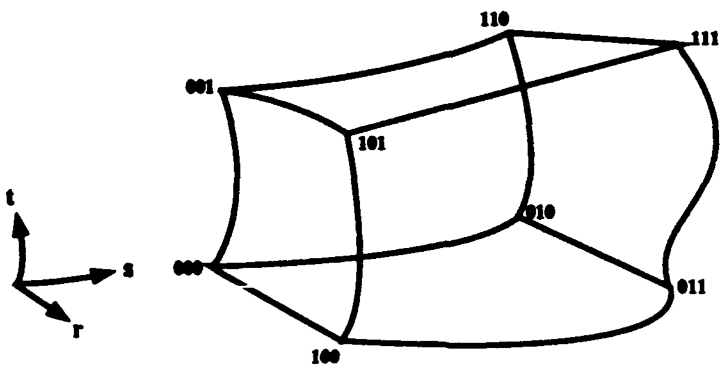


Figure 3.4: *The six sided volume is defined by the twelve bounding curves joining the corner nodes at positions* $\mathbf{x}_{ijk}, i, j, k = 0$ *or 1*

3.4.2 Interpolation between Specified Surfaces

In some circumstances, one or more of the surfaces may be specified, e.g. by some analytic formula. Let us suppose that $\mathbf{X}_i(s,t)$, $s, t\epsilon[0,1]$ represent opposite surfaces passing through the four points $\mathbf{x}_{0jk}$ for $i = 0$ and $\mathbf{x}_{1jk}$ for $i = 1$. Similarly $\mathbf{Y}_j(r,t)$ are surfaces passing through the sets of points $\mathbf{x}_{i0k}$ and $\mathbf{x}_{i1k}$, and $\mathbf{Z}_k(r,s)$ are surfaces through $\mathbf{x}_{ij0}$ and $\mathbf{x}_{ij1}$, respectively.

In the case where all eight surfaces are specified, the relevant interpolant is [4]:

$$\begin{aligned}\mathbf{x}(r,s,t) &= \mathbf{X}_i(s,t)\phi_i(r) + \mathbf{Y}_j(r,t)\psi_j(s) + \mathbf{Z}_k(r,s)\eta_k(t) \\ &- \mathbf{X}_{jk}(r)\psi_j(s)\eta_k(t) - \mathbf{Y}_{ik}(s)\phi_i(r)\eta_k(t) - \mathbf{Z}_{ij}(t)\phi_i(r)\psi_j(s) \\ &+ \mathbf{x}_{ijk}\phi_i(r)\psi_j(s)\eta_k(t) \qquad (3.4.2)\end{aligned}$$

If one or more of the surfaces is not given in this form, but only in terms of its bounding curves, then we use the two-dimensional blending interpolant to give an expression that may be substituted in Eq.(3.4.2). For example, we may set (for $k = 0$ and for $k = 1$),

$$\mathbf{Z}_k(r,s) = \mathbf{X}_{jk}(r)\psi_j(s) + \mathbf{Y}_{ik}(s)\phi_i(r) - \mathbf{x}_{ijk}\phi_i(r)\psi_j(s) \qquad (3.4.3)$$

Re-expressing all of $\mathbf{X}_i$, $\mathbf{Y}_j$ and $\mathbf{Z}_k$ in this way gives us back Eq.(3.4.1).

3.5 Multiblock Decomposition

The multiblock decomposition divides the object into blocks, and these blocks map naturally onto processes on a MIMD computer architecture. Within each block, finite element contributions are assembled on the uniform lattice in curved coordinate space, and particle positions are stored in curved space.

This, as will be shown in later chapters, leads to field equations, charge assignment and current assignment which are the same in all coordinate systems, *and* require no more computational work than for the uniform cartesian mesh case. Geometrical complications are swept into the constitutive relationships and the particle accelerations.

Efficient MIMD software balances work across processors whilst minimising the amount of global data and interprocessor message passing. The physics of Maxwell's equations provide a natural ordering of the data to achieve this, in that interactions at a point in space only involve information from its light cone in space-time; the computational domain is subdivided into blocks. The blocking is chosen to simplify implementation of boundary condition and of data visualisation, and to optimise the utilisation of computer resources. Spatially blocked field data only requires data from neighbouring blocks in the distributed memory MIMD implementation.The master control program only requires information about the block surfaces ("glue patches") which stick together the volume filling meshes in each of the slave block processors. This arrangement offers the prospects of a larger computational intensity and weak Amdahl limit on parallel processor speedup. Moreover, the simple logical square (cube in 3-D) addressing within each block leads to fast serial processing within each slave process.

Without compromising the subdivision of the spatial mesh into blocks to get efficient MIMD processing, one can further demand that boundary conditions *only* apply at the surfaces of block. This completely eliminates the addressing problems in embedding surfaces within blocks, and allows surface data to be passed to the control program through the "glue patch" tables. Further saving of computer storage and time arise from only keeping metric information and material property data in those blocks where they are needed. When a large number of small blocks are needed to describe a complex object, load balancing is achieved by assigning several blocks to one processor.

In summary the mesh (or more correctly, the finite element net) will use:

- A multiblock spatial decomposition where segments of target surfaces and other boundaries are coincident with block surfaces.
- Indirect ("glue patch") addressing between blocks, and logical space (i,j) (or cube (i,j,k) in 3-D) element nets within blocks.
- Transfinite interpolation subdivision of the curvilinear quadrilateral (hexahedral in 3-D) multiblocks into finite elements.

Figure 3.5 illustrates the multiblock decomposition of a complex object into a set of curvilinear sub-block connected by glue patches. The cylinder (a) is divided into five block (b). These blocks are treated as separate objects with their own coordinate systems, and communicate by passing field and particle data via the glue patches (denoted by dashed lines in (c)). (d) the curvilinear quadrilateral (hexahedral in 3-D) blocks are meshed by transforming them to

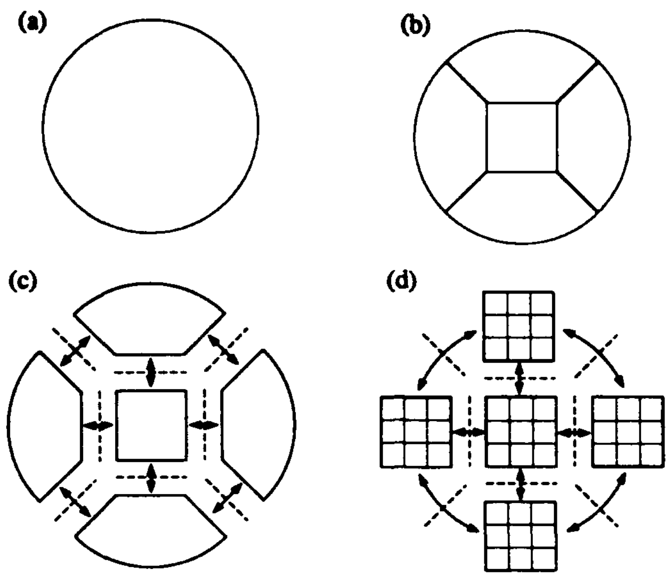


Figure 3.5: *An illustration of the multiblock decomposition and meshing of a complex object.*

rectangles (bricks in 3-D) in curved space and dividing the rectangles (bricks) into unit side square (cube) elements. *Uniblock* particle and field primitives will be defined for the sub-blocks.

Metric and basis vectors are defined by the transformation between the physical and curved space meshes. The simplicity of the computation of metric elements from boundary data in transfinite interpolation offers the option of recomputing values of elements as required from the boundary curve set:- for an $N \times N \times N$ mesh, it reduces necessary storage to the $12N$ boundary vectors! Another alternative, and one we propose adopting, is to store global reference points for each uniblock, plus the covariant basis vectors and metric tensor elements for each block type. In most cases, symmetries and congruences greatly reduce the amount of geometric data to be stored. For instance, the example shown in Figure 3.5 extended to 3-D perpendicular to the plane shown has two block types only; the central rectangular brick block type, which requires only six numbers to completely specify its basis vectors and metric tensor elements, and the surrounding wedge shaped block type, which requires a single plane of values each for the basis vector and metrics. Specifying the four reference points $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ as shown in Figure 3.4 allows similar blocks to be fitted to block types by translation, rotation, reflection and linear scaling.

Chapter 4

Field Equations

In this Chapter, a number of alternative Virtual Particle schemes are derived for the solution of the electromagnetic field equations. Their relative merits are discussed and the particular variant which will form the basis of the MIMD software is specified.

4.1 Introduction

When a virtual particle electromagnetic particle-mesh scheme is derived, it implies a specific method for solving Maxwell's Equations. Such a method is optimal in the sense that it minimises the approximated action, and will inherit the many desirable properties of virtual particle particle-mesh schemes where applicable, e.g. being well suited to parallel computer implementation.

In this chapter, following references [6, 9], virtual particle electromagnetic particle-mesh schemes for general three-dimensional co-ordinate systems are outlined. Effectively the schemes differ only in the forms taken by the constitutive relations. Nevertheless, a meaningful comparison of their relative merits is possible and is set out. The chosen algorithm is summarised at the end of the chapter; We propose that the simplest lumped scheme form the basis of the software in the first instance.

4.2 The Variational Formulation

The electromagnetic field action integral may be written in general curvilinear coordinates $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$ as

$$I = \int dt\, d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3 \sqrt{g} \left\{ \frac{1}{2}(E_i\, D^i - H_i\, B^i) + j^i\, A_i - \rho\phi \right\}, \qquad (4.2.1)$$

where electric field

$$E_k = -\frac{\partial \phi}{\partial \bar{x}^k} - \dot{A}_k, \qquad (4.2.2)$$

magnetic field

$$B^i = \frac{e^{ijk}}{\sqrt{g}} \frac{\partial A_k}{\partial \bar{x}^j}, \tag{4.2.3}$$

electric displacement

$$D^i = \epsilon_o E^i = \epsilon_o g^{ij} E_j, \tag{4.2.4}$$

magnetic intensity

$$H_i = \frac{B_i}{\mu_o} = \frac{1}{\mu_o} g_{ij} B^j, \tag{4.2.5}$$

current density

$$j^i = \sum_p \frac{q_p}{\sqrt{g}} \delta(\bar{x}^1 - \bar{x}^1_p)\delta(\bar{x}^2 - \bar{x}^2_p)\delta(\bar{x}^3 - \bar{x}^3_p)\dot{\bar{x}}^i, \tag{4.2.6}$$

and charge density

$$\rho = \sum_p \frac{q_p}{\sqrt{g}} \delta(\bar{x}^1 - \bar{x}^1_p)\delta(\bar{x}^2 - \bar{x}^2_p)\delta(\bar{x}^3 - \bar{x}^3_p). \tag{4.2.7}$$

The sum over p is over particles, each with charge q_p. The metric tensor elements g_{ij} can be computed from the relationship between the reference cartesian coordinates x^i and the general curvilinear coordinates $\bar{x}^i$, and then $g = \| g_{ij} \|$

Discrete equations are derived by introducing finite element approximations to the potentials

$$\phi = \phi U \tag{4.2.8}$$

$$A_i = \mathsf{A}_{(i)} W_{(i)} \tag{4.2.9}$$

where Φ and $\mathsf{A}_{(i)}$ are nodal amplitudes. Equation (4.2.8) is shorthand for

$$\phi(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = \sum \Phi(\mathbf{k}, n) U(\mathbf{k}, n; \bar{x}^1, \bar{x}^2, \bar{x}^3, t) \tag{4.2.10}$$

where the sum is over spatial (**k**) and temporal (n) node indices. Eq.(4.2.1) is then varied with respect to the nodal amplitudes yielding discrete approximations to the inhomogeneous Maxwell's equations. (The homogeneous Maxwell equations follow by virtue of Eqs.(4.2.2) and (4.2.3), if U and W_i are appropriately chosen).

4.2.1 The Pointwise Scheme

The simplest general geometry extension of the derivation given in reference [6] is given by supposing that the relations

$$d^i = \epsilon_0 \sqrt{g} g^{ij} E_j, \tag{4.2.11}$$

$$H_i = \frac{1}{\mu_0} \frac{g_{ij}}{\sqrt{g}} b^j, \tag{4.2.12}$$

hold in a pointwise sense. Introducing the basis functions V_k for E_k and X_i for $b^i = \sqrt{g}B^i$ implied by substituting Eqs.(4.2.8), (4.2.9) in Eqs.(4.2.2) and (4.2.3) gives nodal equations

$$\partial_i \mathsf{d}^i - \rho = 0, \tag{4.2.13}$$

$$\partial_t \mathsf{d}^i - e^{ijk}\partial_j \mathsf{H}_k + \mathsf{I}^i = 0. \tag{4.2.14}$$

where ∂_j is the centred finite difference operator in the $\bar{x}^j$ coordinate direction, and ∂_t is the centered time difference i.e. if the nodal interval is unity, then

$$\partial_t f(t) = f(t+1/2) - f(t-1/2), \text{ etc.}$$

The homogeneous Maxwell's equations become

$$\partial_i \mathsf{b}^i = 0, \tag{4.2.15}$$

$$\partial_t \mathsf{b}^i + e^{ijk}\partial_j \mathsf{E}_k = 0, \tag{4.2.16}$$

resulting in a need to satisfy constitutive relations

$$\mathsf{d}^i = \epsilon_0 \left\{ \int dt \sqrt{g} d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3\, g^{(i)j} V_{(i)} V_j \right\} \mathsf{E}_j \tag{4.2.17}$$

$$\mathsf{H}_k = \frac{1}{\mu_0} \left\{ \int dt\, d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3 \frac{g_{(k)l}}{\sqrt{g}} X_{(k)} X_l \right\} \mathsf{b}^l. \tag{4.2.18}$$

A disadvantage of this approach is that in view of Eqs.(4.2.14) and (4.2.16), b^i and d^i are natural variables to update, but Eq.(4.2.17) is not explicit for E_i in terms of d^i. To understand the seriousness of the problem, it is sufficient to consider the two-dimensional case, with g^{ij} constant. The lowest order conforming basis functions compatible with charge conservation yield from Eq.(4.2.17), assuming unit mesh-spacing, that

$$\mathsf{d}^x = \epsilon_0 \left\{ g^{xx} \left[\frac{1}{6} \left(\left[\partial_+^x\right]^2 + \left[\partial_-^x\right]^2 \right) \frac{2}{3} I \right] \mathsf{E}_x + g^{xy} \left[\frac{1}{2} (\partial_+^y + \partial_-^y) \right] \mathsf{E}_y \right\}, \tag{4.2.19}$$

and similarly for the y and z components. $\partial_\pm^a$ denotes a shift of plus or minus half a mesh-spacing in the a^{th} co-ordinate. The g^{xy} term is not susceptible to any but the grossest approximation. Taking Eq.(4.2.19) as it stands leads a coupled system of equations involving all nodal values of E_x and E_y. Hence other approaches are investigated in the following sections.

4.2.2 Optimal Schemes

These schemes are derived by replacing the pointwise relations Eqs.(4.2.11) and (4.2.12) with weak approximations that are optimal in a least squares' sense. The replacements do not change the current and charge density terms, hence attention focuses on

$$I_1 = \frac{1}{2} \int dt\, d\bar{x}^1\, d\bar{x}^2\, d\bar{x}^3 \left(\frac{g_{ij}}{\sqrt{g}} \frac{d^i d^j}{\epsilon_0} - \frac{b^i b^j}{\mu_0} \right) \tag{4.2.20}$$

or

$$I_2 = \frac{1}{2}\int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \frac{1}{2}\sqrt{g} g_{ij}\left(\frac{D^i D^j}{\epsilon_0} - \frac{B^i B^j}{\mu_0}\right). \tag{4.2.21}$$

It is convenient to introduce here the inner product notation

$$\langle a, b\rangle = \int_D dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 a(\bar{x}^i, t) b(\bar{x}^i, t), \tag{4.2.22}$$

where D is the domain of interest.

To treat δI_1, Eqs.(4.2.11) and (4.2.12) are rearranged in the forms

$$E_i = \frac{1}{\epsilon_0\sqrt{g}} g_{ij} d^j, \tag{4.2.23}$$

$$H_i = \frac{1}{\mu_0\sqrt{g}} g_{ij} b^j. \tag{4.2.24}$$

Introduce the representation $d^i = \bar{d}^{(i)} V_{(i)}$, then the optimal approximation to Eq.(4.2.23) yields $E^i = \bar{E}^{(i)} V_{(i)}$ where

$$\langle \bar{E}_i V_{(i)}, V_{(i)}\rangle = \langle \frac{1}{\epsilon_0\sqrt{g}} g_{ij}\bar{d}^j V_{(j)}, V_{(i)}\rangle. \tag{4.2.25}$$

Similarly Eq.(4.2.24) gives

$$\langle \bar{H}_i X_{(i)}, X_{(i)}\rangle = \langle \frac{1}{\mu_0\sqrt{g}} g_{ij}\bar{b}^j X_{(j)}, X_{(i)}\rangle. \tag{4.2.26}$$

Varying (4.2.20) yields

$$\delta I_1 = \int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \frac{g_{ij}}{\sqrt{g}}\left(\bar{d}^i \delta\bar{d}^j V_{(i)} V_{(j)} - \bar{b}^i \delta\bar{b}^j X_{(i)} X_{(j)}\right). \tag{4.2.27}$$

Hence on varying Eq.(4.2.25), and inserting it and Eq.(4.2.26) into Eq.(4.2.27);

$$\delta I_1 = \int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \left(\bar{d}^i \delta\bar{E}_i V_{(i)} V_{(i)} - \bar{H}_i \delta\bar{b}^i X_{(i)} X_{(i)}\right). \tag{4.2.28}$$

$\delta\bar{E}_i V_{(i)}$ and $\delta\bar{b}^i X_{(i)}$ can be straightforwardly calculated for variations of Φ and A_i yielding

$$\frac{\delta I}{\delta\Phi} = \int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \left\{-\bar{d}^i \frac{\partial U}{\partial\bar{x}^i} V_{(i)} - qU\right\} = 0 \tag{4.2.29}$$

$$\frac{\delta I}{\delta\mathsf{A}_i} = \int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \left\{-\bar{d}^i \frac{\partial W_{(i)}}{\partial t} V_{(i)} - \bar{H}_j e^{jm(i)} \frac{\partial W_{(i)}}{\partial\bar{x}^m} X_{(j)} + I^{(i)} W_{(i)}\right\} = 0. \tag{4.2.30}$$

The consistency relations

$$\frac{\partial U}{\partial\bar{x}^k} = -\partial_{(k)} V_{(k)}, \tag{4.2.31}$$

$$\frac{\partial W_k}{\partial t} = -\partial_t V_k, \tag{4.2.32}$$

$$\frac{\partial W_i}{\partial \bar{x}^j} = -\partial_j X_k; \;\; ijk \text{ cyclic permutations of } (123), \tag{4.2.33}$$

and the lumping approximations $\langle V_{(i)},\, V_{(i)}\rangle = 1$, $\langle X_{(i)},\, X_{(i)}\rangle = 1$ then bring Eqs.(4.2.29) and (4.2.30) to the form of Eqs.(4.2.13-4.2.14). Note that the left-hand sides of Eqs.(4.2.25) and (4.2.26) have to be lumped, otherwise even for the lowest order conforming elements, these are not explicit formulae for $\bar{E}_i$ and $\bar{H}_i$. A relatively straightforward refinement to the lumped scheme is to use simple iterative schemes to improve the approximation. The diagonal dominance of the mass matrices should lead to rapid convergence.

δI_2 can also be brought to the form (Eqs.(4.2.13-4.2.14) if Eqs. Equations (4.2.11) and (4.2.12) are put into the weak forms

$$\langle \sqrt{g} E_i - \frac{\sqrt{g} g_{ij} D^j}{\epsilon_0},\, V_{(i)}\rangle = 0, \tag{4.2.34}$$

$$\langle \sqrt{g} H_i - \frac{\sqrt{g} g_{ij} B^j}{\mu_0},\, X_{(i)}\rangle = 0. \tag{4.2.35}$$

and the discrete coefficients $\mathbf{d}^i$, $\mathbf{b}^i$ are introduced so that

$$\mathbf{d}^i = \langle \sqrt{g} D^i,\, V_{(i)}\rangle, \tag{4.2.36}$$

$$\mathbf{b}^i = \langle \sqrt{g} B^i,\, X_{(i)}\rangle. \tag{4.2.37}$$

Thus $\mathbf{d}^i$ and $\mathbf{b}^i$ are updated, then the coefficients $\mathbf{D}^i$ and $\mathbf{B}^i$ ($D^i = \mathbf{D}^{(i)} V_{(i)}$, $B^i = \mathbf{B}^{(i)} X_{(i)}$) follow explicitly from Eqs.(4.2.36) and (4.2.37) if the lumping approximation is made, and finally Eqs.(4.2.34) and (4.2.35) yield $\mathbf{E}_i$ and $\mathbf{H}_i$.

4.2.3 Hybrid Schemes

It is possible to derive explicit schemes by keeping the point constitutive relation Eq.(4.2.12) for the magnetic field and the optimal approximation Eq.(4.2.25) or Eq.(4.2.34) for the electric field. We shall not pursue this option further here.

4.3 Properties of the Schemes

In this section the lowest order conforming, charge conserving elements are considered. It will also be assumed that g_{ij} and $\sqrt{g}$ are constant everywhere, corresponding to the case where physical space is split into congruent parallelepipeds. Under these circumstances it is convenient to introduce the operators

$$M^a = \frac{1}{6}\left(\left[\partial_+^a\right]^2 + \left[\partial_-^a\right]^2\right) + \frac{2}{3} I, \tag{4.3.1}$$

$$A^a = \frac{1}{2}\left(\partial_+^a + \partial_-^a\right), \qquad (4.3.2)$$

where $\partial_\pm^a$ were defined following Eq.(4.2.19). The inner products of basis functions may be written simply as

$$\langle V_i, V_j\rangle = \begin{cases} M^b M^c & (i=j=a) \\ A^a A^b M^c & (i=a, j=b) \end{cases}, \qquad (4.3.3)$$

$$\langle X_i, X_j\rangle = \begin{cases} M^a M^t & (i=j=a) \\ A^a A^b M^t & (i=a, j=b) \end{cases}, \qquad (4.3.4)$$

where a, b and c are a permutation of the spatial indices (123) and t denotes time. Henceforth we lump in time, i.e. set $M^t = 1$.

For the pointwise scheme, the constitutive relations take the forms

$$\mathsf{d}^1 = \epsilon_0\sqrt{g}\left(g^{11}M^2M^3\mathsf{E}_1 + g^{12}A^1A^2M^3\mathsf{E}_2 + g^{13}A^1M^2A^3\mathsf{E}_3\right) \qquad (4.3.5)$$

$$\mathsf{H}_1 = \frac{1}{\mu_0\sqrt{g}}\left(g_{11}M^1\mathsf{b}^1 + g_{12}A^1A^2\mathsf{b}^2 + g_{13}A^1A^3\mathsf{b}^3\right) \qquad (4.3.6)$$

Without lumping, the optimal equations for I_1 are in variables $\tilde{d}^i$, $\tilde{b}^i$ where e.g.

$$\tilde{d}^1 = M^2M^3\bar{d}^1,\ \tilde{b}^1 = M^1b^1 \qquad (4.3.7)$$

and the constitutive relations are e.g.

$$M^2M^3\bar{E}_1 = \frac{1}{\epsilon_0\sqrt{g}}\left(g_{11}M^2M^3\bar{d}^1 + g_{12}A^1A^2M^3\bar{d}^2 + g_{13}A^1A^3M^2\bar{d}^3\right) \qquad (4.3.8)$$

$$M^1\bar{H}_1 = \frac{1}{\mu_0\sqrt{g}}\left(g_{11}M^1\bar{b}^1 + g_{12}A^1A^2\bar{b}^2 + g_{13}A^1A^3\bar{b}^3\right). \qquad (4.3.9)$$

For I_2 we find that $\mathbf{E}_i$ is related to $\mathbf{d}^i$ in the same way that $\bar{E}_i$ depends on $\tilde{d}^i$ and similarly for the magnetic field, i.e. when the g_{ij} are global constants, the optimal I_1 and I_2 schemes are identical.

Comparing Eqs.(4.3.6) and (4.3.9) shows that the pointwise relation for H_i is equivalent to the optimal one after lumping in space, i.e. setting $M^i = 1$. The effect of lumping is different, in Eq.(4.3.6) it actually sharpens H_i, whereas in Eq.(4.3.9) the off-diagonal terms are smoothed. In the absence of spatial lumping the pointwise scheme should produce a smoother H_i and might be preferred. However the implicitness of Eq.(4.3.5) means that only a hybrid scheme would be practicable, and if the $M^i = 1$ such a scheme is indistinguishable from the fully optimal ones. The difference between the I_1 and I_2 schemes is slight; the absence of a division by $\sqrt{g}$ may be an advantage for I_2 when the elements are very distorted. However if such situations are avoided, I_1 is preferable because its operation count is likely to be smaller.

4.4 Maxwell's Equations

The principal difference between the cartesian and general curvilinear derivations of the field equations from the action integral (4.2.1) is the need for approximations to evaluate Eqs.(4.2.4),(4.2.5) and metric tensor elements in the latter case. Evaluation of the metric elements was dealt with in Section 3.2. Choices of approximations to Eqs.(4.2.4) and (4.2.5) lead to implicitness in computing either E_i from d^j for the circulation term in Faraday's Law or d^j from E_i in applying boundary conditions; the latter is favoured for computational simplicity, particularly since the boundary condition is implicit only for non-orthogonal element nets at external boundaries. The scheme we shall implement will be the I_1 optimal scheme, with (at least in the first instance) lumped 'mass' matrices.

All of the Virtual Particle schemes discussed in Section 4.2 yield the approximations to Maxwell's Equations:

$$\partial_t \mathsf{b}^i = -e^{ijk}\partial_j \mathsf{E}_k, \quad \partial_i \mathsf{b}^i = 0 \tag{4.4.10}$$

$$\partial_t \mathsf{d}^i = e^{ijk}\partial_j \mathsf{H}_k - I^i, \quad \partial_i \mathsf{d}^i = Q \tag{4.4.11}$$

The precise relationship between quantities b^i,d^i, E_k,H_k and the nodal amplitudes depends on the approximations chosen for the constitutive relationships. In the lowest order lumped approximation to the I_1 scheme they become the nodal amplitudes.

The analysis leading to Eqs.(4.4.10) and (4.4.11) shows that the discrete equations in the quantities b^i,d^i, E_k,H_k,I^i and Q are identical in any coordinate system. Geometrical information appears, along with the permeability and permittivity tensors, only in the constitutive relations relating d^i to E_i and b^i to H_i.

Figure 4.1 shows the location of the nodes that arises from the choice of linear potential representation in three dimensions. Note that the forms of the 'difference' equations Eqs.(4.4.10) and (4.4.11) are *identical* to those given by the Yee algorithm [13] on a uniform cartesian net. They differ from the Yee algorithm in the variables appearing in the equations, and the element net is not necessarily rectangular in physical space.

4.4.1 2-D General Geometry Example

Field equations for the virtual particle scheme for quadrilateral elements in 2-D are discussed in this section. As for the cartesian case considered in [6], we shall focus on the case of linear support. We assume that the general coordinate $\bar{x}^3$ is ignorable, i.e. potential and field derivatives with respect to $\bar{x}^3$ are zero.

All quadrilateral elements are assumed (without loss of generality) to map onto unit square elements in the curved $(\bar{x}^1, \bar{x}^2)$ coordinate space. If we assume mixed linear/constant support over the elements in this curved space, as was

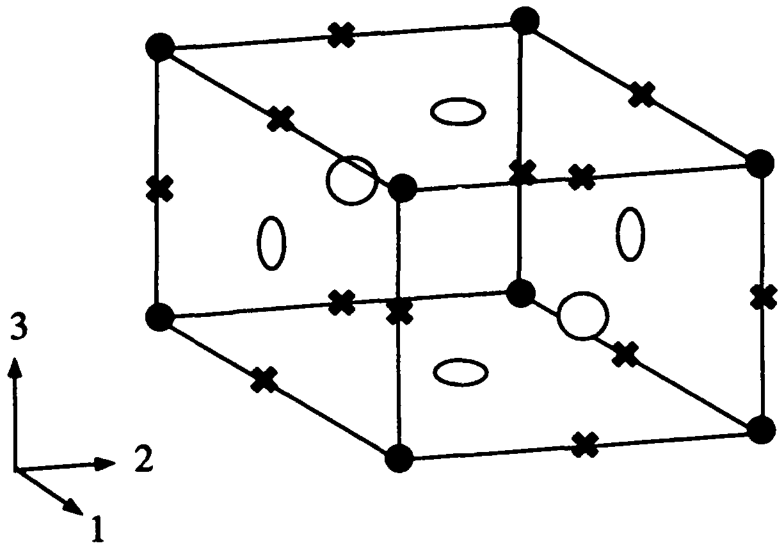


Figure 4.1: *The location of nodes on the unit cube element. Crosses give E_i, I^i and d^i node locations, open circles give H_i and b^i locations and solid circles give position and scalar potential node locations.*

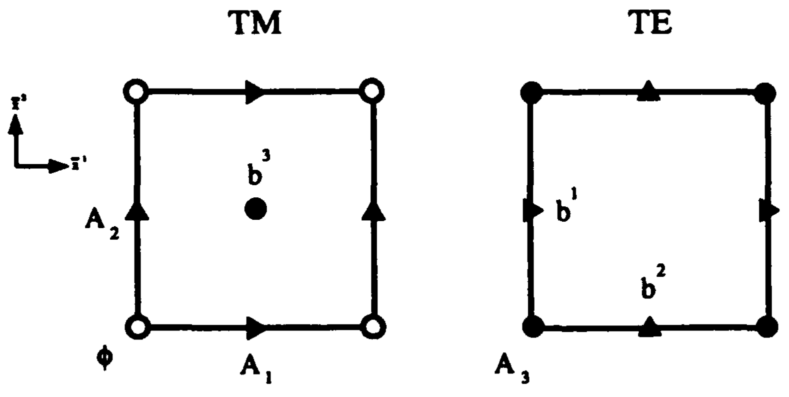


Figure 4.2: *The location of scalar potential, Φ, vector potential (A_1, A_2, A_3) and magnetic field (b^1, b^2, b^3) nodal amplitudes. Fields H_k have same spatial locations as b^k and E_i, d^i and I^i have same spatial locations as A_i*

used for the 2-D cartesian example in [6]), then the location of nodal values on a single elements will be as illustrated in Fig 4.2. These nodes are interlaced in time. Located at half-integral timelevels are values of A_i, b^i, H_i, I^i and at integral timelevels, nodal values of E_i, d^i, Φ and q are defined.

It follows from the results of Reference [6] that the assembled finite element node equations can be written in operator form:

Initial conditions:-

$$\partial_1 \mathsf{d}^1 + \partial_2 \mathsf{d}^2 = \rho \tag{4.4.12}$$

$$\partial_1 \mathsf{b}^1 + \partial_2 \mathsf{b}^2 = 0 \tag{4.4.13}$$

Transverse Magnetic (TM) equations

$$\partial_t \mathsf{d}^1 = \partial_2 \mathsf{H}_3 - \mathsf{I}^1 \tag{4.4.14}$$

$$\partial_t \mathsf{d}^2 = -\partial_1 \mathsf{H}_3 - \mathsf{I}^2 \tag{4.4.15}$$

$$\partial_t \mathsf{b}^3 = \partial_2 \mathsf{E}_1 - \partial_1 \mathsf{E}_2 \tag{4.4.16}$$

Transverse Electric (TE) equations

$$\partial_t \mathsf{b}^1 = -\partial_2 \mathsf{E}_3 \tag{4.4.17}$$

$$\partial_t \mathsf{b}^2 = \partial_1 \mathsf{E}_3 \tag{4.4.18}$$

$$\partial_t \mathsf{d}^3 = \partial_1 \mathsf{H}_2 - \partial_2 \mathsf{H}_1 - \mathsf{I}^3 \tag{4.4.19}$$

4.5 Constitutive Equations

The weak approximations Eqs.(4.2.25) and (4.2.26) to Eqs.(4.2.4) and (4.2.5), with lumped mass matrices give the simplest explicit expressions for E_i and H_i. Their evaluation gives tensor equations

$$\mathsf{E}_i = \mathsf{G}^E_{ij}\mathsf{d}^j \tag{4.5.1}$$

$$\mathsf{H}_i = \mathsf{G}^H_{ij}\mathsf{b}^j \tag{4.5.2}$$

where elements of the symmetric tensors G^E_{ij} and G^H_{ij} are sparse matrices.

The lowest order lumped approximations lead to matrices G_{ii} which are diagonal, and matrices G_{ij} with four nonzero elements per row. Figure 4.3 illustrates this for

$$\mathsf{E}_2 = \mathsf{G}^E_{21}\mathsf{d}^1 + \mathsf{G}^E_{22}\mathsf{d}^2 \tag{4.5.3}$$

E_2 is computed at the central node in Figure 4.3 from d^2 and G^E_{22} at that node, and from d^1 at the four neighbouring nodes as indicated by arrows. In general, different values of G^E_{21} are associated with each arrow. In three dimensions, $\mathsf{G}^E_{23}\mathsf{d}^3$ gives similar contributions from the four neighbouring nodes at which d^3 are stored.

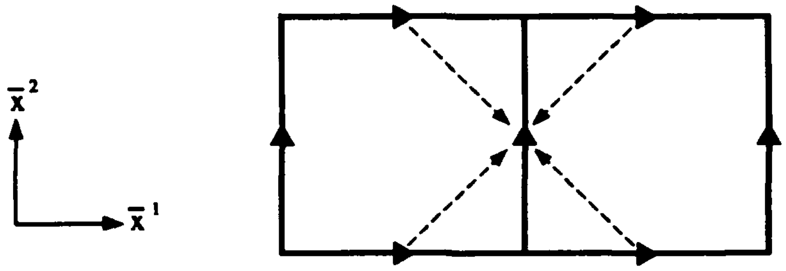


Figure 4.3: *Computation of* E_2 *at the central node using the lowest order lumped approximation to Eq.(4.5.1) involves values of* d^2 *at that node, and from* d^1 *at the four neighbouring nodes as indicated by arrows.*

4.6 Algorithm Summary

Four different schemes for solving Maxwell's equations have been derived, and their relative merits examined. All have the desirable attributes of being charge conserving, having good dispersive properties and being well suited to parallel computer implementation. For the situation where lumping is a good quality/cost compromise and the finite elements are not too distorted, the optimal I_1 scheme with lumping appears most suitable; it is this scheme that will form the basis of the benchmark program.

Local block coordinates. The multiple decomposition divides the computational space into a set of curvilinear quadrilateral (hexahedral) blocks. Each of these blocks has associated with it a local cartesian coordinate system, which is related to the global cartesian coordinates by some translations and/or rotations. For the purposes of integrating Maxwell's equations for tensor components forward in time, only the metric tensor is required, but for extracting physical components of the field, and for the particle integrations, knowledge of the transformation between cartesian and curved space coordinates is required.

Fixed timestep. All curved space finite elements are chosen to be cubes with unit sides. For the present discussion, we shall further assume that a fixed timestep of unity in the curved space is used, although this assumption can be straightforwardly relaxed.

Dimensionless units. For fixed timestep, it is convenient to introduce dimensionless units for physical space quantities. These imply the scaling factors summarized in Table 4.1 to convert the quantities appearing in Eqs.(4.6.1) - (4.6.8) into SI units. In the table, μ_0, ϵ_0 and c have their usual meanings, Δt is the timestep, E_0 is some reference electric field (to be set to simplify the particle equations) and $Z_0 = \mu_0 c = 377\Omega$ is the impedance of free space.

Coordinate transformations. Given the basis vectors $\mathbf{e}_i$ and $\mathbf{e}^i$ and the local block cartesian with unit vectors are $\hat{\mathbf{x}}_i$, the electric field $\mathbf{E}$ may be

Table 4.1: *Scaling factors to convert quantities to SI units*

Quantity	**Scale factor**
electric field	E_0
magnetic field	E_0/c
length	$c\Delta t$
time	Δt
velocity	c
basis vector ($\mathbf{e}_i$)	$c\Delta t$
reciprocal basis vector ($\mathbf{e}^i$)	$1/c\Delta t$
metric tensor elements (g_{ij})	$(c\Delta t)^2$
A_i, Φ, E_i, b^i	$E_0 c\Delta t^2$
d^i, I^i, Q	$E_0 c\Delta t^2/Z_0$
H_i	$E_0 c\Delta t/Z_0$
permittivity (ϵ)	ϵ_0
permeability (μ)	μ_0

written

$$\mathbf{E} = \tilde{E}_j \hat{\mathbf{x}}_j = E_i \mathbf{e}^i \tag{4.6.1}$$

where $\tilde{E}_i$ are the local cartesian components of the physical electric field, and E_j are the covariant components.

Dotting Eq.(4.6.1) with $\mathbf{e}_j$ gives

$$E_i = (\mathbf{e}_i \cdot \hat{\mathbf{x}}_j)\tilde{E}_j \tag{4.6.2}$$

which may be written in matrix notation $E = T\tilde{E}$, where the 3x3 matrix T has elements $T_{ij} = \mathbf{e}_i \cdot \hat{\mathbf{x}}_j$. Similarly, the physical magnetic intensity $\tilde{H}$ is related to the covariant components H_i by

$$H_i = T_{ij}\tilde{H}_i \tag{4.6.3}$$

The evaluation of $\mathbf{e}_i$ and thence T_{ij} was treated in Section 3.2.

4.6.1 The Timestep Loop

The timestep loop for the integration of the field equations deals exclusively with tensor field components. The input from the particle integration are the contravariant currents I^i, and the output to the particle integration are field components E_i and b^i. Details of the particle integration and boundary condition will be discussed later. The finite element field equations are assembled within each block, and glue patch data exchange is used to complete the assembly at block surfaces. This leads to the following steps in the timestep loop:

1. compute current from particle move
2. obtain H from b in each block

$$\langle X_{(i)}(H_i - \frac{g_{ij}}{\mu\sqrt{g}}b^j)\rangle = 0 \qquad (4.6.4)$$

3. compute displacement currents contributions in each block

$$\dot{d}^i = \langle -H_j e^{jm(i)} \frac{\partial W_{(i)}}{\partial \bar{x}^m} - I^i W_{(i)}\rangle \qquad (4.6.5)$$

4. assemble $\dot{d}^i$ contributions at block surfaces by gluepatch data exchange. This step applies the 'internal' boundary conditions: e.g. periodic, symmetry, neumann and domain decomposition boundary conditions
5. update d^i and apply surface boundary conditions (conductors, resistive walls, external circuit couplings, etc) to d^i. At internal boundaries, the new d^i is computed from

$$\langle -d^{(i)} \frac{\partial W_{(i)}}{\partial t}\rangle = \dot{d}^i \qquad (4.6.6)$$

6. obtain E from d in each block

$$\langle V_i(E_i - \frac{g_{ij}}{\epsilon\sqrt{g}}b^i)\rangle = 0 \qquad (4.6.7)$$

7. assemble E_i contributions across internal boundaries by glue patch data exchange
8. apply surface boundary conditions to E_i
9. advance b^i in each block

$$\frac{\partial b^i}{\partial t} = -e^{ijk}\partial_j E_k \qquad (4.6.8)$$

10. compute particle accelerations

In the above equations, lowest order lumping is used to make the field equations and constitutive relations explicit. Otherwise, iteration or direct solution will lead to further message passing between blocks as d^i, E_i, etc are refined. Items (1), (4) and (10) are dealt with in more details in the next two chapters.

Chapter 5

Particle Equations

This chapter presents the Virtual Particle charge and current assignment schemes in general curvilinear coordinates for the linear basis function finite elements discussed in the previous chapter for Maxwell's equations. Assignment is the same in all coordinate systems and has the charge conserving property. Treatment of the equations of motion using curved space coordinates for positions and cartesians for momenta are outlined.

5.1 Variational Formulation

In general curvilinear coordinates $(\bar{x}^1,\ \bar{x}^2,\ \bar{x}^3)$ the action integral may be written

$$I = \int dt\ d\bar{x}^1\ d\bar{x}^2\ d\bar{x}^3 \sqrt{g} \left\{ \frac{1}{2}(E_i\ D^i - H_i\ B^i) + j^i\ A_i - \rho\phi \right\} + I_K \qquad (5.1.1)$$

where I_K is the kinetic lagrangian. If we further assume that the distribution function is represented by a set of sample points (ie 'superparticles'), then the source terms in the field lagrangian become

$$j^i = \sum_p \frac{q_p}{\sqrt{g}} \delta(\bar{x}^1 - \bar{x}^1_p)\delta(\bar{x}^2 - \bar{x}^2_p)\delta(\bar{x}^3 - \bar{x}^3_p)\dot{\bar{x}}^i \qquad (5.1.2)$$

$$\rho = \sum_p \frac{q_p}{\sqrt{g}} \delta(\bar{x}^1 - \bar{x}^1_p)\delta(\bar{x}^2 - \bar{x}^2_p)\delta(\bar{x}^3 - \bar{x}^3_p) \qquad (5.1.3)$$

and the kinetic lagrangian term becomes

$$I_K = -\int dt\ \sum_p \frac{Mc^2}{\gamma_p} \qquad (5.1.4)$$

The sums in p are over particles, each with charge q_p.

Treating I as a functional of the vector potential A_i, the scalar potential ϕ and particle coordinates $\{\mathbf{x}_p\}$ led to Euler-Lagrange equations representing

Maxwell's equations; as shown in the previous chapter, the discrete approximations to these differential equations are obtained by substituting test function approximations for ϕ, A_i and $\mathbf{x}_p$ and taking variations with respect to the nodal amplitudes.

The use of sample points ('superparticles') reduces the velocity space integrals to sums over particles, and transforms the Vlasov equation to the relativistic equations of motion for the superparticles. Source terms, Eqs.(5.1.2) and (5.1.3), arise from variations with respect to potentials in the differential limit. These variations give the charge and current assignment schemes in the finite element case, as illustrated in Sections 5.2 and 5.3. Section 5.2.1 shows that the assignment has the charge conserving property even for non-orthogonal element nets. In the differential limit, variation of the action with respect to particle coordinates gives the relativistic momentum equation.

The treatment of the equations of motion is outlined in Section 5.4 In principle, the equations of motion can be obtained from Eq.(5.1.1) by considering variations with respect to particle positions. In practice, this has proved too arduous except in the limit of infinitessimal timestep, and so recourse has been taken to the conventional finite difference methods to discretise the equations of motion in time.

5.2 2-D Assignment

The finite element test function approximations to the potentials may be written $\phi = \Phi U$ and $A_i = A_{(i)}W_{(i)}$, where Φ and A are nodal amplitudes, and sums are implied over element nodes (cf Ref [6]).

Variation of Eq.(5.1.1) with respect to Φ yield charge assignment:

$$Q = \int dt \sum_p q_p U((\bar{x}_p^1,\ \bar{x}_p^2,\ \bar{x}_p^3)(t), t) \tag{5.2.5}$$

Similarly, variation with respect to A gives current assignment:

$$\mathsf{I}^i = \int dt \sum_p q_p \dot{\bar{x}}_p^{(i)} W_{(i)}((\bar{x}_p^1,\ \bar{x}_p^2,\ \bar{x}_p^3)(t), t) \tag{5.2.6}$$

The integrals for Q and I^i are evaluated in exactly the same manner as described in Ref [6]. For example, if in a general quadrilateral element U and W_i have the same linear / piecewise dependence as their cartesian counterpart [6, Section 4.3], then the fractions of the charge assigned from a charge q_p at position $(\bar{x}^1, \bar{x}^2)$ (marked by open circles) to the four nodes at the corners of the element as shown in Figure 5.1(a) are given by

$$\begin{array}{ll} \delta Q_{NW} = q_p(1-\bar{x}^1)\bar{x}^2 & \delta Q_{NE} = q_p\bar{x}^1\bar{x}^2 \\ \delta Q_{SE} = q_p\bar{x}^1(1-\bar{x}^2) & \delta Q_{SW} = q_p(1-\bar{x}^1)(1-\bar{x}^2) \end{array} \tag{5.2.7}$$

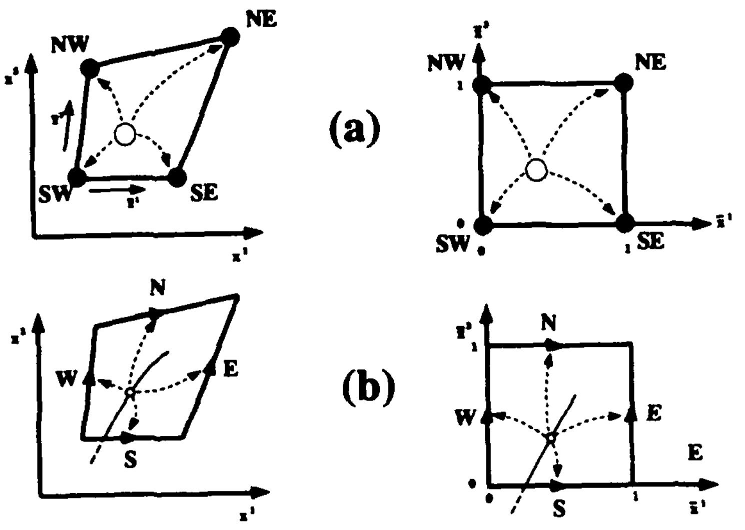


Figure 5.1: *(a) Charge assignment from a single particle and (b) current assignment from a single particle trajectory segment for linear potential test functions. The figures at the left and right are respectively physical and curved space representations.*

Similarly, evaluating Eq.(5.2.6) for the transverse magnetic (TM) contravariant current components from a trajectory segment from $(\bar{x}_i^1, \bar{x}_i^2)$ to $(\bar{x}_f^1, \bar{x}_f^2)$ yields

$$\begin{array}{ll} \delta \mathsf{I}_N = q_p \bar{X}^2 \Delta \bar{x}^1 & \delta \mathsf{I}_S = q_p(1 - \bar{X}^2)\Delta \bar{x}^1 \\ \delta \mathsf{I}_E = q_p \bar{X}^1 \Delta \bar{x}^2 & \delta \mathsf{I}_W = q_p(1 - \bar{X}^1)\Delta \bar{x}^2 \end{array} \qquad (5.2.8)$$

where $\bar{X}^k = (\bar{x}_f^k + \bar{x}_i^k)/2$, $\Delta \bar{x}^k = \bar{x}_f^k - \bar{x}_i^k$, where k=1 or 2. Coordinates $(\bar{x}_i^k, \bar{x}_i^k)$ and $(\bar{x}_f^k, \bar{x}_f^k)$ are respectively the start and end the trajectory segment (solid line in Figure 5.1(b)) in the finite element. Trajectory segments are assumed to be straight lines in the curved $(\bar{x}^1, \bar{x}^2)$ space. Figure 5.1(b) shows the position of the virtual particle at $(\bar{X}^k, \bar{X}^k)$ as a small open circle on the trajectory segment and labels the location of the current nodes referred to in Eq.(5.2.8).

The $\bar{x}_3$ direction current components leads to terms quadratic in the path parameter, and will so lead to two virtual particles being required to give the trajectory segment contributions to the currents at the four corner nodes. Figure 5.2 shows the virtual particle locations for the TM and TE current components from the same trajectory segment in curved space. The I^3 current require contributions from two virtual particles, at positions

$$(\bar{X}^1, \bar{X}^2) \pm (\Delta \bar{x}^1,\ \Delta \bar{x}^2)/2\sqrt{3},$$

both with strengths $\Delta \bar{x}^3/2$. Summing the contributions of the two virtual

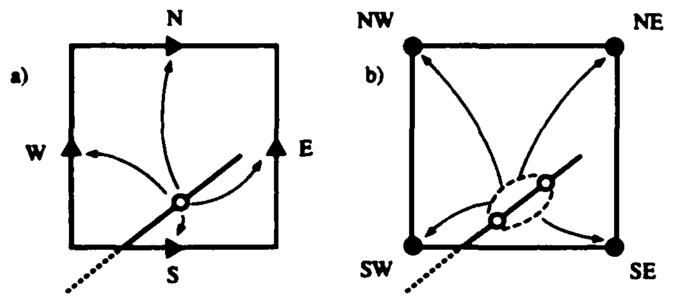


Figure 5.2: *Current assignment a) for TM current components and b) for the TE (= I^3) current component. The open circles show the locations of the virtual particles on the trajectory segment (solid line) lying in the current element.*

particles gives

$$\delta\mathsf{I}_{SW} = \Delta\bar{x}^3\left[(1-\bar{X}^1)(1-\bar{X}^2)+\frac{\Delta\bar{x}^1\Delta\bar{x}^2}{12}\right] \tag{5.2.9}$$

$$\delta\mathsf{I}_{NW} = \Delta\bar{x}^3\left[(1-\bar{X}^1)\bar{X}^2-\frac{\Delta\bar{x}^1\Delta\bar{x}^2}{12}\right] \tag{5.2.10}$$

$$\delta\mathsf{I}_{NE} = \Delta\bar{x}^3\left[\bar{X}^1\bar{X}^2+\frac{\Delta\bar{x}^1\Delta\bar{x}^2}{12}\right] \tag{5.2.11}$$

$$\delta\mathsf{I}_{SE} = \Delta\bar{x}^3\left[\bar{X}^1(1-\bar{X}^2)-\frac{\Delta\bar{x}^1\Delta\bar{x}^2}{12}\right] \tag{5.2.12}$$

5.2.1 Charge Conservation

Charge and current assignment are linear operations, so by linear superposition, conservation for one trajectory moving through a single time step implies the same for the sum of all trajectories. Summing all contributions to Eqs.(5.2.8) from a single particle, gives the same as the difference of Eqs.(5.2.7) at start and end points, ie the linear quadrilateral (hexahedral in three dimensions) element case satisfies

$$\partial_t Q = -\partial_k \mathsf{I}^k \tag{5.2.13}$$

where the symbol ∂ denotes a centred difference arising from assembling the finite element contributions, and Q and I^k are nodal amplitudes. Equations of the form (5.2.13) can be shown to be generally satisfied for VP algorithms.

5.3 3-D Assignment

5.3.1 Charge

If we label nodes on the unit cube element (i, j, k) and write the assignment fun ion in terms of its product parts

$$U_{ijk}(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = w_i(\bar{x}^1) w_j(\bar{x}^2) w_k(\bar{x}^3) \tag{5.3.1}$$

From Eq.(5.2.5), the charge assigned to element node (i, j, k) where $i, j, k = 0$ or 1 from a charge q_p becomes

$$Q_{ijk} = q_p w_i(\bar{x}^1) w_j(\bar{x}^2) w_k(\bar{x}^3) \tag{5.3.2}$$

where

$$w_0(x) = 1 - x \;\; ; \;\; w_1(x) = x \tag{5.3.3}$$

$(\bar{x}^1, \bar{x}^2, \bar{x}^3)$ is the displacement of the particle position from the element corner $(0, 0, 0)$.

5.3.2 Current

Using the impulse approximation to particle motion as described in [6], and the factorisation of W reduces, Eq.(5.2.6) to the contribution of a single particle to

$$\mathsf{I}^1_{\frac{1}{2}jk} = \int_{-\frac{1}{2}}^{\frac{1}{2}} ds q_p \Delta\bar{x}^1 w_j(\bar{X}^2 + s\Delta\bar{x}^2) w_k(\bar{X}^3 + s\Delta\bar{x}^3) \tag{5.3.4}$$

The quadratic integral is evaluated exactly by point Gaussian quadrature, so Eq.(5.3.4) can be written as the contribution from virtual particles at $\mathbf{x}_\pm$:-

$$\mathsf{I}^1_{\frac{1}{2}jk} = q_p \frac{\delta\bar{x}^1}{2} \left[w_j(\bar{x}^2_+) w_k(\bar{x}^3_+) + w_j(\bar{x}^2_-) w_k(\bar{x}^3_-) \right] \tag{5.3.5}$$

where $\Delta\bar{x}^i = \bar{x}^i_f - \bar{x}^i_i$ are components of the length of the trajectory segment in the current element, and $\bar{X}^i = (\bar{x}^i_f + \bar{x}^i_i)/2$ are the components of the mean position. Indices j and k take values 0 or 1. The positions of the virtual particle have coordinates

$$\bar{x}^i_\pm = \bar{X}^i \pm \frac{\Delta\bar{x}^i}{2\sqrt{3}} \tag{5.3.6}$$

The corresponding expressions for current components $\mathsf{I}^2_{k\frac{1}{2}j}$ and $\mathsf{I}^3_{jk\frac{1}{2}}$ are given by simultaneously cyclically rotating component indices (123) and node indices $(\frac{1}{2}jk)$.

Figure 5.3 gives a cartesian space sketch of assignment according according to a) Eq.(5.3.2) for charge and b) Eq.(5.3.5) for current.

In practice, it more convenient to evaluate explicitly the integrals for I^i rather than use quadrature. Equation (5.3.5) then becomes

$$\mathsf{I}^1_{\frac{1}{2}jk} = q_p \Delta\bar{x}^1 \left[w_j(\bar{X}^2) w_k(\bar{X}^3) + (j - \frac{1}{2})(k - \frac{1}{2}) \frac{\Delta\bar{x}^2 \Delta\bar{x}^3}{3} \right] \tag{5.3.7}$$

and similarly for I^2 and I^3 by cyclic rotation of indices.

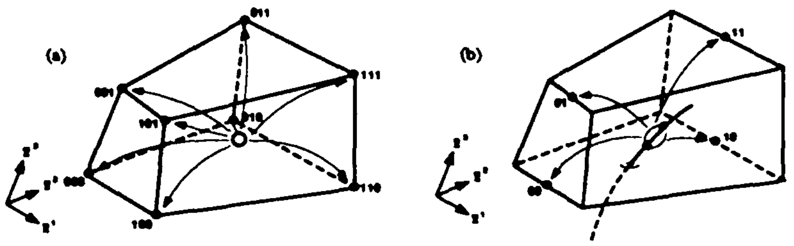


Figure 5.3: *(a) Charge assignment from a single particle and (b)* I_1 *current assignment from a single particle trajectory segment for linear potential test functions in physical space.*

5.4 Particle Motion

5.4.1 Positions

The change in particle position is computed in the same manner as the cartesian case. The change of position is computed from the leapfrog approximation

$$\bar{x}^k(n+1) = \bar{x}^k(n) + \bar{v}^k(n+1/2) \tag{5.4.1}$$

to

$$\frac{d\bar{x}^k}{dt} = \bar{v}^k \tag{5.4.2}$$

where argument n is used to denote timelevel.

The evaluation of Eq.(5.4.1) requires the computation of $\bar{v}^k$ from the momentum. To avoid the explicit treatment of the Christoffel symbol in the momentum equation, momenta are stored in terms of their local cartesian components. These components are in units of m_0c, where m_0 is the rest mass of the species in question.

Given the cartesian momenta $\mathbf{p} = \hat{p}_i\hat{\mathbf{x}}^i$, the contravariant components are given by

$$\hat{p}_i = \hat{\mathbf{x}}^i.\mathbf{e}_j\bar{p}^j \tag{5.4.3}$$

or in matrix form

$$\hat{p} = T^T\bar{p} \tag{5.4.4}$$

The relativistic gamma is given by

$$\gamma = \sqrt{1+\hat{p}_i\hat{p}_i} \tag{5.4.5}$$

Hence Eq.(5.4.1) may be written in matrix form

$$\bar{x}(n+1) = \bar{x}(n) + (\mathrm{T}^T)^{-1}\hat{p}/\gamma \tag{5.4.6}$$

Note that Eq.(5.4.6) is not properly time centred unless T is evaluated at the half timelevel, in which case the equation becomes implicit. In the first instance, we shall use the explicit approximation, although this can be refined by using some simple predictor-corrector on Eq.(5.4.6), or by solving the cartesian space form of Eq.(5.4.1) and transforming the cartesian coordinate to curved space.

5.4.2 Momenta

The finite element approximation in space and time for the particle accelerations leads to unwieldy expressions; consequently, we have replaced the time derivative by a finite difference. Treating the momentum equation in curved space introduces the Christoffel symbol Γ^l_{km} in the quadratic centripetal term:

$$\frac{d\bar{p}_k}{dt} - \Gamma^l_{km}\bar{v}^m\bar{p}_l = F_k = q(E_k + e_{klm}\bar{v}^l b^m) \tag{5.4.7}$$

A more straightforward approach is to use the cartesian leapfrog scheme for the momentum

$$\frac{d\mathbf{p}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \tag{5.4.8}$$

The local cartesian components of the electric and magnetic fields can be found using the contravariant basis vectors. Using matrix notation and T as defined in Eq.(5.4.4), the cartesian electric field is related to the covariant components by

$$\hat{E} = \mathsf{T}^{-1}\bar{E} \tag{5.4.9}$$

and the cartesian magnetic field components are given by

$$\hat{B} = \mathsf{T}^T\bar{b}/det(\mathsf{T}) \tag{5.4.10}$$

With the choice of field units

$$E_0 = \frac{2m_0c}{q\Delta t} \tag{5.4.11}$$

the resulting leapfrog approximation to Eq.(5.4.8) becomes

$$\mathbf{p}(n+1/2) - \mathbf{p}(n-1/2) = 2\mathbf{E} + (\mathbf{p}(n+1/2) + \mathbf{p}(n-1/2)) \times \mathbf{B}/\gamma \tag{5.4.12}$$

which is readily solved using standard methods [12, Chap 4].

Chapter 6

Boundary Conditions

This chapter summarises the application of interior and exterior boundary conditions to both the field equations and the particle equations.

6.1 Introduction

The field boundary conditions fall into two classes: interior and exterior. Interior boundary conditions are those where the computational domain is extended to complete the assembly of Ampere's equation at boundary nodes. Instances of the interior boundary conditions are periodic, Neumann (symmetry) and glue patch boundaries. Their application involves adding some 'external' contribution to $\partial_t d^i$ from elements of the same form as those in the body of the computational domain. Interior field boundary conditions are treated in Section 6.2.

Exterior field boundary conditions (Section 6.3) couple the Ampere's equation for boundary nodes to external circuit equations relating surface currents to surface fields. The simplest external circuit is the perfect conductor, where tangential electric fields are zero. If nonzero tangential fields are applied at the conducting wall boundary, then one obtains the fixed applied field boundary condition. Slightly more involved is the resistive wall, which differs from the perfect conductor in that tangential electric and magnetic fields are related by a nonzero surface impedance. More general external circuit couplings at boundaries, involving inductance and capacitance lead to differential equations relating surface electric fields to surface currents.

Particle boundary conditions may be likewise classed as interior or exterior. Interior conditions follow the same classification as the fields, and exterior conditions are either particle emission or absorption. These are treated in more detail in Section 6.4

6.2 Fields at Interior Boundaries

For all cases, the application of interior field boundary conditions involves assembling contributions to $\dot{d}$ (Eq.(4.6.5)) across adjacent block boundaries. This assembly is performed by adding in the gluepatch contributions. Figure 6.1 illustrates the gluepatch data transfer. Data stored on the logical (i, j, k) lattice of nodes in block 1 are copied to the gluepatch, and then data is added from the gluepatch to the corresponding node on block 2. On both source and target blocks are reference points (solid circles and crosses in Figure 6.1) which specify the location and orientation of the blocks with respect to the glue patch. The reference points are used to perform the field component and indexing transformations between blocks.

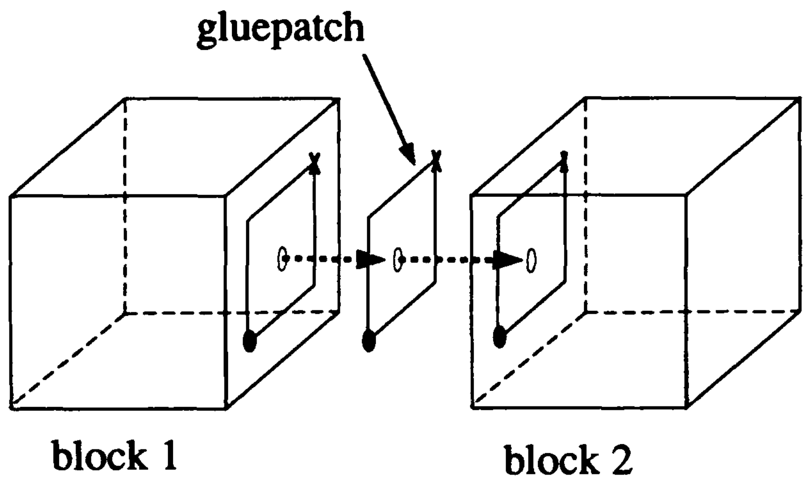


Figure 6.1: *All interior boundary conditions are applied by passing data between blocks via gluepatches.*

Two types of field gluepatch are required:

Surface patch to exchange the two tangential field components with nodes common to the faces of two blocks.

Line patches to exchange the one tangential field component with nodes common to the edges of four or more blocks.

Interior boundary conditions are applied by adding the gluepatch values $\dot{d}^i_{glue}$ to the corresponding node accumulator

$$\dot{d}^i := \dot{d}^i + \dot{d}^i_{glue} \qquad (6.2.1)$$

A simple example of the use of gluepatches is to apply doubly periodic boundary conditions to a 2-D rectangular domain. Four gluepatches are required: two surfaces patches to connect the north to south boundary and the east to west boundary, and two line patches to connect the NE to SW corner and the SE to NW corner. Gluepatch exchanges are required twice each timestep, once for $\dot{d}^i$ to complete the integration of Ampere's equation, and once for E_i to complete the computation of E from d in evaluating the constitutive relationship.

6.3 Fields at Exterior Boundaries

Exterior boundary conditions are applied to blocks through boundary condition patches, which are stuck onto the blocks in the same manner as the gluepatches, only now they connect to external conditions rather than to another block.

If we let $E_t = -\mathbf{n} \times (\mathbf{n} \times \mathbf{E})$ be the (physical) electric field tangential to the boundary surface whose unit normal is $\mathbf{n}$, then exterior boundary conditions may be summarised as follows:-

$$\mathbf{E}_t = \mathsf{E} \quad : \quad \text{specified applied field} \qquad (6.3.1)$$

$$\mathbf{E}_t = 0 \quad : \quad \text{perfect conductor} \qquad (6.3.2)$$

$$E_t = Z\mathbf{j}_s \quad : \quad \text{resistive wall} \qquad (6.3.3)$$

$$\mathbf{E}_t = \mathbf{E}_t(\mathbf{j}_s) \quad : \quad \text{circuit equation} \qquad (6.3.4)$$

E is the (given) applied field, Z is a given surface impedance, and $\mathbf{j}_s$ (curl $(\mathbf{H})$) is the surface current. The circuit equation case is in general a differential equation in time relating $\mathbf{E}_t$ to $\mathbf{j}_s$ of which Eqs.(6.3.1-6.3.3) are special cases. For the present discussion we shall limit ourselves to the three special cases listed.

The ease with which the boundary conditions can be applied depends on the nature of the finite element not at the boundary surface. We identify four different cases

- orthogonal nets
- boundary orthogonal nets
- surface orthogonal nets
- general curvilinear nets

The first three cases usually lead to local surface equations for the fields which are explicit, whilst the fourth leads to all implicit equations which has either to be solved by iteration, or lumped further to give explicit expressions for displacement fields.

Boundary conditions are applied to Ampere's equation through prescribed (contravariant) displacement fields and/or displacement currents at boundary nodes. Boundary conditions on Faraday's equation are applied by prescribing the corresponding (covariant) electric fields.

We shall consider the application of boundarv conditions to surfaces perpendicular to basis vector $\mathbf{e}^1$, as illustrated in Figure 6.2.

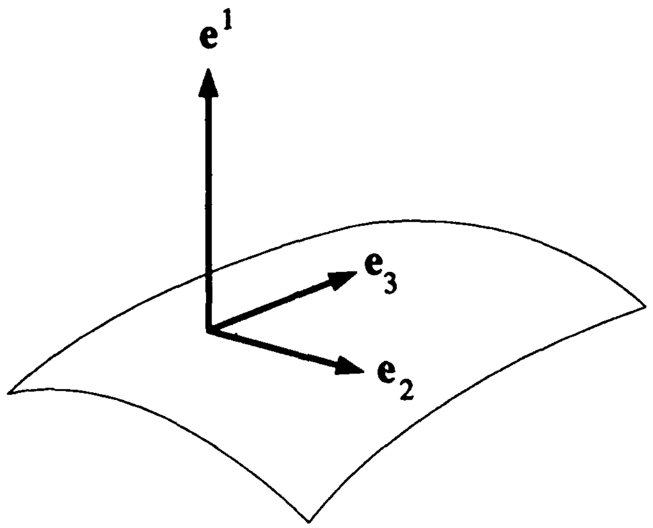


Figure 6.2: *Basis vectors* $\mathbf{e}_2$ *and* $\mathbf{e}_3$ *lie on surface perpendicular to reciprocal basis vector* $\mathbf{e}^1$

6.3.1 Applied Field

The applied field boundary condition sets the surface tangential electric field to some specified value E, so that at the surface.

$$\mathbf{n} \times (\mathsf{E} - \mathbf{E}) = 0 \tag{6.3.5}$$

For the face illustrated in Figure 6.2, $\mathbf{n} = \mathbf{e}^1/|\mathbf{e}^1|$, so Eq.(6.3.5) yields $\mathbf{e}^1 \times (\mathsf{E} - \mathbf{E}) = \mathbf{e}^1 \times \mathbf{e}^i(\mathsf{E}_i - E_i)$

$$= \frac{1}{\sqrt{g}}(\mathbf{e}_3(\mathsf{E}_2 - E_2) - \mathbf{e}_2(\mathsf{E}_3 - E_3)) \tag{6.3.6}$$

Taking the dot product of Eq.(6.3.6) with $\mathbf{e}^2$ and $\mathbf{e}^3$ gives the covariant field component boundary conditions

$$E_2 = \mathsf{E}_2 \;\; ; \;\; E_3 = \mathsf{E}_3 \tag{6.3.7}$$

In the continuum limit, Eq.(6.3.6) can likewise be written in terms of the contravariant fields:

$$d^2 - \frac{\mathsf{E}^2}{\epsilon\sqrt{g}} = d^1 \frac{\mathbf{e}^2 \cdot \mathbf{e}^1}{|\,\mathbf{e}^1\,|^2} \;\; ; \;\; d^3 - \frac{\mathsf{E}^3}{\epsilon\sqrt{g}} = d^1 \frac{\mathbf{e}^3 \cdot \mathbf{e}^1}{|\,e^1\,|^2} \tag{6.3.8}$$

For the boundary orthogonal case, $\mathbf{e}^1 \times \mathbf{e}_1 = 0$, and Eqs.(6.3.8) reduce to simple Dirichlet conditions for d^2 and d^3:

$$d^2 = \frac{\mathsf{E}^2}{\epsilon\sqrt{g}} \; ; \; d^3 = \frac{\mathsf{E}^3}{\epsilon\sqrt{g}} \tag{6.3.9}$$

One possible approach to handling the boundary conditions in the simulation code is to use Eqs.(6.3.8) in a pointwise fashion at boundary nodes. A more consistent approach is to use Eqs.(6.3.7) for the covariant fields, and the lumped finite element constitutive relations to solve for d^2 and d^3 at boundary nodes:-

$$\mathsf{E}_2 = G_{21}d^1 + G_{22}d^2 + G_{23}d^3 \tag{6.3.10}$$

$$\mathsf{E}_3 = G_{31}d^1 + G_{32}d^2 + G_{33}d^3 \tag{6.3.11}$$

For boundary orthogonal elements ($G_{21} = G_{31} = 0$), conditions of the form Eqs. (6.3.9) may again be recovered.

For surface orthogonal elements, $G_{23} = 0$, and Eqs.(6.3.10 and 6.3.11) reduce to

$$d^2 = G_{22}^{-1}(\mathsf{E}_2 - G_{21}d^1) \tag{6.3.12}$$

$$d^3 = G_{33}^{-1}(\mathsf{E}_3 - G_{31}d^1) \tag{6.3.13}$$

In two dimensions (where the 3 coordinate is negligible, say, and $G_{23} = G_{31} = 0$), explicit equations of the form Eqs.(6.3.12) - (6.3.13) can always be found.

For general curvilinears, Eqs.(6.3.10 and (6.3.11) lead to the matrix equations for d^2 and d^3

$$(G_{22} - G_{23}G_{33}^{-1}G_{32})d^2 = (\mathsf{E}_2 - G_{23}G_{33}^{-1}\mathsf{E}_3) + G_{21} - G_{23}G_{33}^{-1}G_{31})d^1 \tag{6.3.14}$$

$$(G_{33} - G_{32}G_{22}^{-1}G_{23})d^3 = (\mathsf{E}_3 - G_{32}G_{22}^{-1}\mathsf{E}_2) + (G_{31} - G_{32}G_{22}^{-1}G_{21})d^1 \tag{6.3.15}$$

which are discrete analogues to Eqs.(6.3.8).

6.3.2 Perfect Conductor

Perfect conductor boundary conditions differ from the applied field case only in that the applied field is zero.

6.3.3 Resistive Wall

The resistive wall boundary condition, applicable for small skin depth surfaces, relates the surface current $\mathbf{j}_s$ to the tangential electric field at the surface:

$$\mathbf{n} \times \mathbf{H} = \mathbf{j}_s \tag{6.3.16}$$

$$-\mathbf{n} \times (\mathbf{n} \times \mathbf{E}) = Z\mathbf{j}_s \tag{6.3.17}$$

For the surface shown in Figure 6.2, $\mathbf{n} = \mathbf{e}^1 / \mid \mathbf{e}^1 \mid$, and Eq.(6.3.17) gives

$$\mathbf{e}^1 \times (\mathbf{E} - Z\mathbf{j}_s) \tag{6.3.18}$$

Following the reasoning given above for the applied field case, the surface covariant fields satisfy

$$E_2 = Zj_{s_2} \ ; \ E_3 = Zj_{s_3} \tag{6.3.19}$$

The corresponding equation to Eq.(6.3.8) is

$$d^2 - \epsilon ZJ^2 = d^1 \frac{\mathbf{e}^2 \cdot \mathbf{e}^1}{\mid e^1 \mid^2} \quad d^3 - \epsilon ZJ^3 = d^1 \frac{\mathbf{e}^3 \cdot \mathbf{e}^1}{\mid e^1 \mid^2} \tag{6.3.20}$$

and Eq.(6.3.14) is replaced by

$$(G_{22} - G_{23}G_{33}^{-1}G_{32})(d^2 - \epsilon ZI^2) = (G_{21} - G_{23}G_{33}^{-1}G_{31})d^1 \tag{6.3.21}$$

The evolutionary equation for d^2 is obtained by using Eq.(6.3.21) to eliminate the surface current, I^2, from the finite element assembled equation for the surface nodes.

6.4 Particles at Interior Boundaries

Each field surface gluepatch has a corresponding particle gluepatch. Particles passing through a gluepatch from a source block to a target block are passed from the storage areas of the source block particle tables to that of the target block via the gluepatch buffer.

Position coordinates are transformed from the curved space components of the source block to those of the target block. Momentum coordinates are converted from the local cartesian coordinates of the source block to those of the target block.

6.5 Particles at Exterior Boundaries

Particles are lost from the simulation domain by absorption at exterior boundaries, and are introduced by emission at boundaries.

Emission is determined by additional physical models. For example:

Thermionic emission, where surface electric fields draw electrons out of a cathode surface boundary layer.

Secondary electrons, whose distribution is determined by impacting particles and boundary material properties.

beam injection, where external sources determine density and momentum distributions.

Chapter 7

Data Organisation

This Chapter outlines the data organisation. A multilevel tabular approach is proposed in order to obtain efficient MIMD implementation whilst maintaining flexibility.

7.1 Introduction

The data organisation problem to be resolved is how to map a multiblock decomposition of a complex microwave device onto a distributed memory MIMD computer such as the Intel iPSC or a Meiko i860/Transputer MIMD computers. The objective is to maximise program portability and flexibility *without* compromising efficiency.

The principal unit of decomposition is the uniblock - which carries both field and particle data from a volume of space. To handle complex shapes, some blocks will need to be small, whereas for simple shapes, physical boundary condition constraints allow large blocks. If desired, large blocks can be subdivided to facilitate mapping onto the MIMD computer. The data organisation must also allow several small blocks to be collected together on a given process to obtain effective load balance.

Section 7.2 illustrates the multiblock decomposition for a coaxial to cylindrical to rectangular waveguide junction. The global and local data organisation required for such a decomposition is treated in Section 7.3. Section 7.4 proposes a data storage within the block which allows metric data compression for blocks with symmetry and orthogonality properties.

7.2 The Mesh

Figure 7.1 shows the multiblock representation of a cylindrical device with an attached waveguide. Each block is subdivided into elements, and the same parameterisation is used for the coordinate transformation as is employed for the scalar potential (cf Chapter 3).

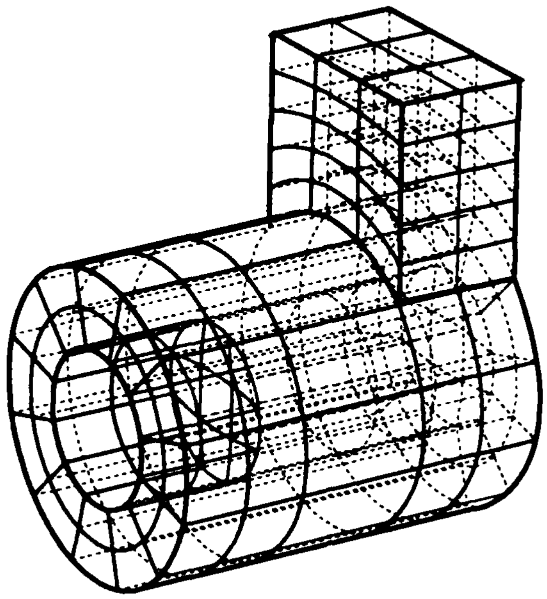


Figure 7.1: *An illustration of the meshing of the junction of a coaxial and rectangular guide.There are six blocks:the rectangular guide, the central cylinder and four in the annulus.*

The multiblock subdivision of the computational domain minimises the amount of global data and interprocessor message passing, and simplifies load balancing across processors. Each slave block only requires data from its neighbours, and the master control program only requires information about the block surfaces ('glue patches') which join the slave blocks together. This arrangement offers the prospects of large computational intensity and a weak Amdahl limit to speedup on distributed memory MIMD computers. Moreover, the simple logical cube addressing within each block leads to fast serial processing.

Without compromising the subdivision of the spatial mesh into blocks to get efficient MIMD processing, one can further demand that boundary conditions *only* apply at the surfaces of blocks; this eliminates the addressing problems in embedding surfaces within blocks, and allows surface data to be passed to the control program through the 'glue patch' tables. Further saving of computer storage and time arise from keeping metric information and material property data only in those blocks where they are needed. When many small blocks are used to describe a complex object, load balancing is achieved by assigning several blocks to one processor.

In summary, the efficient meshing strategy uses

- A multiblock spatial decomposition where segments of target surfaces and other boundaries are coincident with block surfaces.

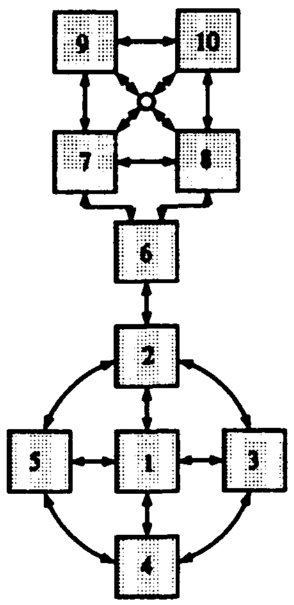


Figure 7.2: *The curved space schematic of the device in Figure 7.1. Blocks transform to rectangular bricks connected by gluepatches (indicated by arrows).*

- Indirect ('glue patch') addressing between blocks, and logical cube (i,j,k) element nets within blocks.

- Transfinite interpolation to divide the curvilinear hexahedral multiblocks into finite elements

Figure 7.2 gives a 'curved space' schematic of the device illustrated in Figure 7.1 with an additional multiblock extension to the rectangular waveguide. The bottom five blocks correspond to the central cylinder and annulus. The arrows indicate block faces connected by gluepatches. Block 6 is the connection from the annulus (block 2) to rectangular guide (blocks 7-10). Blocks 7-10 have been included to illustrate the need for extra line gluepatches for the field solver when more than three block edges meet at a line.

Figure 7.3 summarises the timestep loop operations for a general uniblock. The uniblock (Figure 7.3(a)) has regular mesh of unit spacing (in curved space). Its surface is either covered by boundary patches or is connected to surface glue patches (plus line patches at edges). All boundary conditions and exchanges of particles and fields between uniblocks are handled via the gluepatches and their buffers. If dynamical load balancing is excluded, the only interprocess communication required is the exchange of gluepatch buffer information between pairs of contiguous blocks.

Figure 7.3(b) gives the steps of the timestep loop for the uniblock. The first part gives the updated particle momenta, positions and currents, and

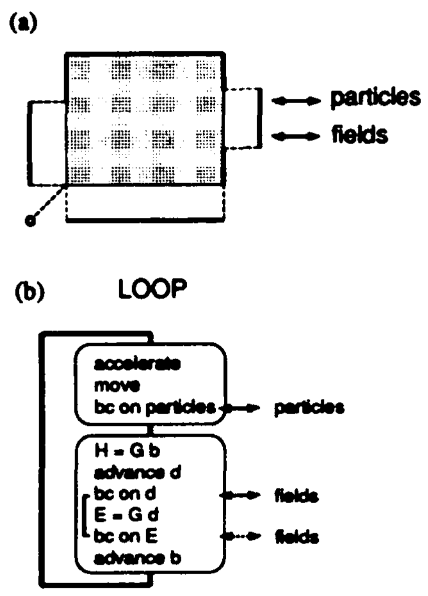


Figure 7.3: *(a) the uniblock surface is covered by boundary or glue patches. (b) the timestep loop for a uniblock.*

exchanges particles with neighbouring blocks and boundaries. This part is complete only when all the particle exchange buffers are empty. The second part is the field solver, where the update of Ampere's equation leads to the exchange of displacement fields, and the update of Faraday's equation leads to the exchange of electric fields. The inner loop over the boundary conditions is relevant only where necessary for the three dimensional general geometry case.

7.3 Data Addressing

Figure 7.4 summarises the global addressing and division of data in global and local. The MIMD virtual particle simulation program treats a microwave devices as a collection of uniblocks connected together by a network of gluepatches. Each gluepatch is glue to two connecting block faces (or block edges in the case of line patches), so that the complete network can be described by stating the positioning of the two faces of each gluepatch on the uniblocks.

There are four levels in the global addressing of this network: the **processor, process, block** and **patch**:

Processor: Since we want the same program to run with the minimum changes on many different parallel computers, a distinction is made between the logical processes used in the program and the physical processors of the

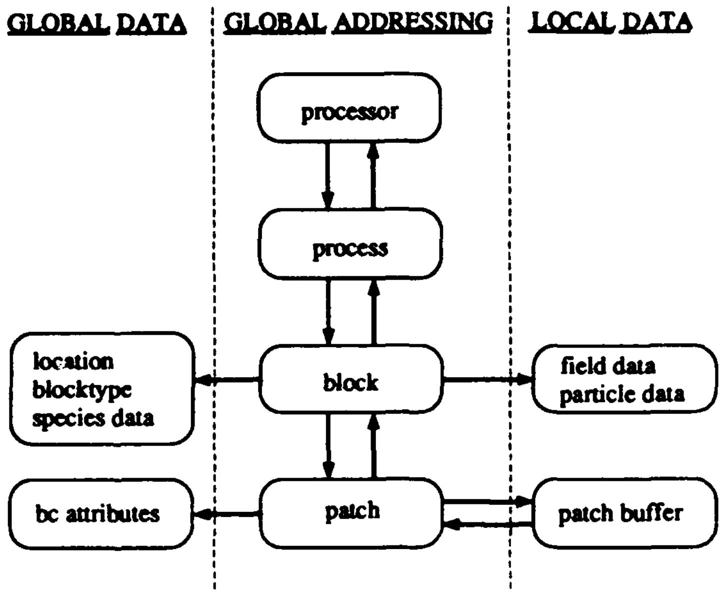


Figure 7.4: *Data Addressing*

MIMD computer. The translation between logical program processes and physical processors can then be kept to the outermost level, where the program is interfaced with the particular parallel computer on which it is to run. Although there is a one-to-one correspondence between a logical process and a physical processor, the numbering of the processors and their physical connectivity (e.g. hypercube or mesh) is highly system-dependent. Subroutines and tables will be provided to translate rapidly between the above logical and physical numbers. By this method we can move to a new MIMD computer by making a few changes to a small number of tables and subroutines, without altering, in any way, the bulk of the complex simulation program. The program is, therefore written in terms of processes which may be thought of as logical processors, and which are translated to physical processor numbers by these tables.

Process: Several blocks may be computed on each processor of a MIMD computer, and the balancing of the computational load between the processors is achieved by moving blocks between processors, until the computational load is roughly equal on all processors. Remembering that the program is to be written in terms of processes, load balancing can be achieved by moving blocks between processes. The distribution of the computation across the parallel computing resource is then described by stating which blocks are on which processes.

Block: The block addressing points to the global block data - its location in space relative to some global coordinates, blocktype and global particle attributes. The block type data will contain metric coefficients needed by the field solver and contravariant basis vectors used by the particle integrator.

Patch: All the exchange of data between blocks will be performed through an exchange patch subroutine. This must recognise whether a target patch to which it is to send data is a patch on a block belonging to its process, or on a block being computed by another process. In the former case, a memory-to-memory copy of the data is performed, and in the latter a message is sent to the target process.

Two way addressing between the four levels is proposed to simplify coding; blocks can be addressed by process and process by blocks, and similarly for processor/process, and block/patch. Direct addressing links are summarised by the arrows in Figure 7.4.

7.4 Uniblock Data Storage

Particle and mesh data in each uniblock will be stored in a similar fashion to that described in [5] Particle coordinates are grouped by species, and mesh data is mapped onto one dimensional arrays.

The nodes for the mesh data are indexed as follows

- Each block of the multiblock decomposition of the computational domain comprises $n_1 \times n_2 \times n_3$ elements.
- The active block domain comprises elements numbered $(0, 0, 0)$ to $(n_1 - 1, n_2 - 1, n_3 - 1)$.
- The nodes in the active block are labelled $(i, j, k) = (0, 0, 0)$ to (n_1, n_2, n_3). Where necessary (ie on the $i = n_1, j = n_2$ and $k = n_3$ planes), an extra padding layer of elements is introduced.
- Optionally, there may be extra bordering layers of elements introduced onto the block. The depth of the border is (lb_1, lb_2, lb_3) in the (i, j, k) directions respectively.

The meshing of the block is illustrated in Figure 7.5. There are $N_\alpha = (n_\alpha + 2lb_\alpha + 1)$ elements and nodes in the α direction.

The total number of data values stored for *each* scalar field (or vector component) is

$$N_1 \times N_2 \times N_3 \qquad (7.4.1)$$

The node locations on each element are sketched in Figure 7.6 and summarised in Table 7.1. The locations are the positions of the nodes for the unit cube element in curved space.

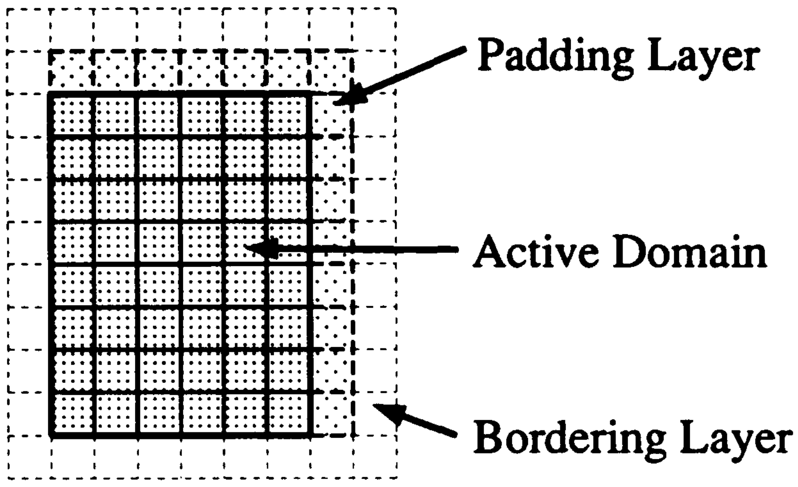


Figure 7.5: *A 2-D illustration of the meshing of a block. The field is solved in the active domain, the padding layer is used to simplify addressing, and provision for a bordering layer is made for possible future use in particle mesh applications.*

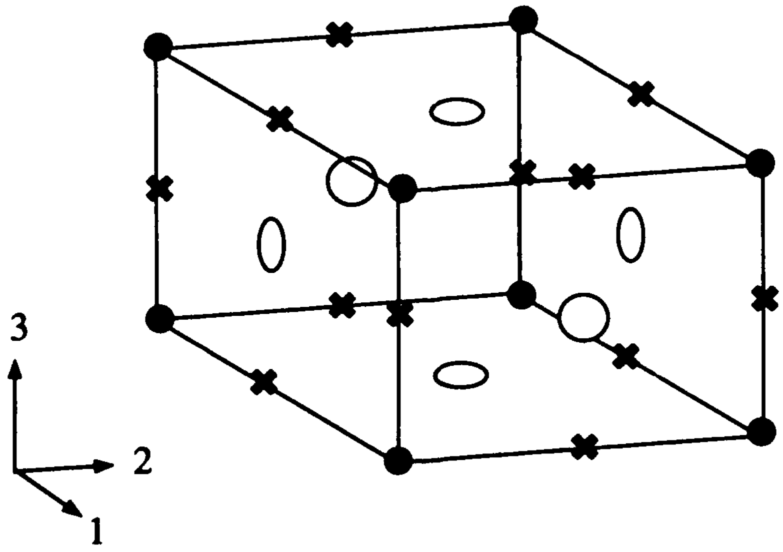


Figure 7.6: *The location of nodes on the unit cube element. Crosses give E_i, I^i and d^i node locations, open circles give H_i and b^i locations and solid circles give position and scalar potential node locations.*

Table 7.1: *Node locations and active index ranges*

components			locations	active index range of nodes		
E_1	d^1	(I^1)	$[\frac{1}{2}00](\frac{1}{2}01)(\frac{1}{2}10)(\frac{1}{2}11)$	$(0, n_1 - 1)$	$(0, n_2)$	$(0, n_3)$
E_2	d^2	(I^2)	$[0\frac{1}{2}0](0\frac{1}{2}1)(1\frac{1}{2}0)(1\frac{1}{2}1)$	$(0, n_1)$	$(0, n_2 - 1)$	$(0, n_3)$
E_3	d^3	(I^3)	$[00\frac{1}{2}](01\frac{1}{2})(10\frac{1}{2})(11\frac{1}{2})$	$(0, n_1)$	$(0, n_2)$	$(0, n_3 - 1)$
H_1	b^1		$[0\frac{1}{2}\frac{1}{2}](1\frac{1}{2}\frac{1}{2})$	$(0, n_1)$	$(0, n_2 - 1)$	$(0_1 n_3 - 1)$
H_2	b^2		$[\frac{1}{2}0\frac{1}{2}](\frac{1}{2}1\frac{1}{2})$	$(0, n_1 - 1)$	$(0, n_2)$	$(0, n_3 - 1)$
H_3	b^3		$[\frac{1}{2}\frac{1}{2}0](\frac{1}{2}\frac{1}{2}1)$	$(0, n_1 - 1)$	$(0, n_2 - 1)$	$(0, n_3)$
$\mathbf{e}_i$			$[000](100)(010)(001)$ $(111)(011)(101)(110)$	$(0, n_1 - 1)$	$(0, n_2 - 1)$	$(0, n_3 - 1)$

Only those nodes with the smallest indices are deemed to belong to the element; in the Fortran implementation these nodes have the same indexing as the element. In Table 7.1, the node belonging to the current element is shown in square braces [...], the remaining locations are nodes on its boundaries belonging to neighbouring elements - hence the use of a padding layer.

To simplify the coding of the cyclic interchange of indices and of compressed storage of basis vectors and metrics, the multidimensional elements described above will be mapped onto one dimensional arrays in the Fortran coding. The mapping will be performed as follows:-

- By (arbitrary) choice, the components of a vector field are stored as three successive scalar fields. (This enforces $D = 1$ below).

- All three dimensional scalar fields are explicitly mapped onto one dimensional arrays as follows:-

  Element i, j, k is stored in location

$$loc(ijk) = \Omega + i \times l_1 + j \times l_2 + k \times l_3 \tag{7.4.2}$$

where

$$\begin{aligned} \Omega &= \textit{mesh origin} \\ l_1 &= D = \text{dimension (1 for scalar, 3 for triplet, etc)} \\ l_2 &= l_1 \times N_1 \\ l_3 &= l_2 \times N_2 \\ N_\alpha &= n_\alpha + 1 + 2lb_\alpha \end{aligned}$$

The origin is chosen so that (i, j, k) corresponds to the element with its corner at the origin of the active domain. Thus, to avoid overwriting before the first element of the array the origin Ω must satisfy

$$\Omega \geq \Omega_{min} = 1 + lb_1 \times l_1 + lb_2 \times l_2 + lb_3 \times l_3 \tag{7.4.3}$$

- If vectors are stored as successive components, then the successive mesh origins are at

$$\begin{aligned} &\Omega_1 \\ \Omega_2 &= \Omega_1 + N_1N_2N_3 \\ \Omega_3 &= \Omega_2 + N_1N_2N_3 \end{aligned}$$

The advantages of this data storage method are

1. it allows the *same* code to be used for 1, 2 and 3 dimensional cases.
2. the same code is applied to all three components by using an outer loop which cyclically loops through components.
3. by defining separately the increments for the metric tensor components (and for basis vectors in the particle acceleration computation), compressed storage of metrics can be employed.

The storage advantage of the last item becomes apparent when comparing storage of G_{11}, say, for the extreme cases of a uniform cartesian block and a general curvilinear block in 3-D: The former requires one floating point number to be stored, whilst the latter requires $n_1 \times n_2 \times n_3$.

Bibliography

[1] W Arter and J W Eastwood, "Electromagnetic Modelling in Arbitrary Geometries by the Virtual Particle Particle-Mesh Method", Paper PWE15, *Proc 14th Int Conf Num Sim Plasmas*, APS, Annapolis (Sep 1991).

[2] N J Brealey, "Computer Simulation of Helix Travelling Wave Tubes", *Proc High Power Microwave generation and Applications*, Varenna, Italy (Sep 1991).

[3] A D Burns, N S Wilkes, I P Jones and J R Kightley, "FLOW3D: body-fitted coordinates", AERE-R12262 (1986).

[4] A D Burns and N S Wilkes, "A finite difference method for the computation of fluid flows in complex three dimensional geometries", AERE-R12342.

[5] J W Eastwood, R W Hockney and D N Lawrence, "P3M3DP - The Three Dimensional Periodic Particle-Particle / Particle-Mesh Program", *Computer Phys Commun* **19**, (1980) 215-261.

[6] J W Eastwood, "The Virtual Particle Electromagnetic Particle-Mesh Method", Comput. Phys. Commun. 64 (1991) 252-266.

[7] J W Eastwood, pp 655-660, Proc. 7th Ann Rev Prog App Computational Electromagnetics, Naval Postgraduate School, Monterey, Ca (Mar 1991).

[8] J W Eastwood, "Computer Modelling of Microwave Sources", *Proc 18th Annual Conf Plasma Phys*, IoP, Colchester (July 1991).

[9] J W Eastwood and W Arter, "Electromagnetic PIC Modelling in Arbitrary Geometry", 1992 ACES Symposium, Monterey, California, March 1992.

[10] W J Gordon, SIAM J. Numer. Anal. 8 (1971) 158.

[11] W J Gordon and C A Hall, Int. J. Numer. Meths. Engng. 7 (1973) 461.

[12] R W Hockney and J W Eastwood, "*Computer Simulation Using Particles*", (Adam Hilger/IOP Publishing, Bristol & New York, 1988).

[13] K S Yee, "Numerical Solution of Initial Boundary Value Problems involving Maxwell's Equations in Isotropic Media", *IEEE Trans Antennas Propagat* **14** (1966) 302-307.

C Annex 2: mimdpic program .doc files

```
C
C
C
C
C             **********************
C             *                    *
C             *    m i m d p i c   *
C             *                    *
C             **********************
C
C
C             2-d MIMD PIC Test Program
C
C             BY
C
C             James W EASTWOOD and Roger W HOCKNEY
C
C
C                                 AEA Technology
C                                 Culham Laboratory
C                                 Abingdon
C                                 Oxon OX14 3DB
C                                 England
C
C    This is a preliminary version of the program
C
C      (c) UNITED KINGDOM ATOMIC ENERGY AUTHORITY - 1992
C
C
C
C
C
C
C
C
C
C    VERSION  1.00.00   JWE/RWH   Culham Laboratory   May 1992
C
```

INDSUB.doc

```
C-----------------------------------------------------------------------
CL                          INDEX OF SUBPROGRAMS
C
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C
CL                           CONTROL             CLASS 0
C  EXPERT        test data dump version                        0.4
C  RECORD        Read and write restart coordinates            0.5
C  BASIC         Version without input file read for MIMD      0.1Z
C  MODIFY        Version to create separate ouput files        0.2Z
C
CL                           PROLOGUE            CLASS 1
C  LABRUN        Label the run                                 1.1
C  CLEAR         Clear common arrays and variables             1.2
C  PRESET        Set default values, not MYPRES etc            1.3
C  AUXVAL        Compute auxillary values                      1.5
C  RESUME        resume from previous run                      1.7
C  START         start the calculation                         1.8
C  RECTBK        set up rectangular brick blocktype            1.10
C  SETDMP        setup tables for xpatch                       1.11
C  NODASG        assign processes to processors(nodes)         1.12
C  PINIT         initialise particles                          1.13
C  G2LMAT        find global to local coord transformation matr 1.14
C  XACMAT        find acceleration transformation matrices     1.15
C  NILVEC        set block vector to zero                      1.20
C  SETVEC        set block vector to constant vector           1.21
C  DATA          Input data specific to run                    1.4R
C  INITAL        Define geometrical and physical IV            1.6B
C
CL                           CALCULATION         CLASS 2
C  STEPON        step on the calculation                       2.1
C  SETCUR        Initialise current arrays                     2.10
C  MOVCUR        Move particles and compute currents           2.11
C  ACCEL         Update particle momenta                       2.12
C  EMITEL        Emit electrons from cathode surfaces          2.13
C  PARTIO        load particles into/from exchange buffers     2.14
C  PPSORT        sort particles into patch buffer              2.15
C  PCXFRM        transform coord to target block coords        2.16
C  PBLINK        manage particle buffering between blocks      2.17
C  QSHARE        assign charge to block mesh                   2.18
C  INJECT        apply particle injection bc                   2.19
C  AMPERE        Compute displacement current                  2.20
C  FARADA        Advance magnetic field one step               2.21
C  GBTOH         Compute H = Gb                                2.22
C  GDTOE         Compute E = Gd                                2.23
C  BCSURD        apply bc on d at surfaces                     2.24
C  BCOPAT        apply orthogonal bc on d on patches           2.25
C  ADDVEC        add two vectors in block together             2.26
C  BCSURE        apply bc on E at surfaces                     2.27
C  BCONE         apply orthogonal bc on E on patches           2.28
C  BCSYM         apply symmetry bc                             2.29
C  GLUEIO        transfer data to/from gluepatch buffers       2.30
C  XPATCH        exchange gluepatch buffers                    2.31
C  BLKIO         copy data between block and gluepatch buffers 2.32
C  XPART         exchange particle coordinates                 2.33
C  CPYVEC        copy block vector                             2.34
C  AVEVEC        average two vectors in block                  2.35
C
CL                           OUTPUT              CLASS 3
C  OUTPUT        Control the output                            3.1
C  GLOBLT        print GLOBal Link Table                       3.2
C  PATOP         patch diagnostic output                       3.3
C  FRAMEL        labelled frame advance                        3.4
C  OP3VEC        print vector                                  3.5
C  PARTOP        particle table summary                        3.6
C  SCAPLT        particle scatter plot                         3.7
C  MSHSUM        summary of mesh storage                       3.8
C  PAKSUM        summary of particle storage                   3.9
C  GRDPLT        plot parallel projection of block grid        3.10
C  GHINIT        Initialise and title graphical output         3.11
C  NETPLT        plot finite element net                       3.12
C  LINEGR        plot line graph of net field                  3.13
C  OPBUF         particle patch exchange buffer op             3.14
C
CL                           EPILOGUE            CLASS 4
C  TESEND        Test for completion of run                    4.1
C  ENDRUN        Terminate the run                             4.2
C
CL                           DIAGNOSTICS         CLASS 5
C  CLIST         Dump common variables                         5.2
C  ARRAYS        Dump common arrays                            5.3
C
CL                           UTILITIES           CLASS P
C  OLYFOR        olympus support routines                      P.1
C  CSEND         dummies for Intel routines                    P.1
C
```

INDCOM.doc

```
C---------------------------------------------------------------------
CL                          INDEX OF COMMON BLOCKS
C
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C
CL                 1.       GENERAL OLYMPUS DATA
C  CHABAS          Basic system character parameters          C1.1
C  COMBAS          Basic system parameters                    C1.2
C  COMIMD          MIMD Related addressing                    C1.8
C  COMDDP          Development and diagnostic parameters      C1.9
C
CL                 2.       PHYSICAL PROBLEM
C  COMPCN          Physical CoNstants                         C2.1
C  COMIBC          physical Iv and BC                         C2.2
C  COMFLD          FieLD arrays                               C2.3
C  COMMET          METric arrays                              C2.4
C  COMPAR          PARticle array                             C2.5
C
CL                 3.       NUMERICAL SCHEME
C  COMNUM          NUMerical constants                        C3.1
C  COMBUF          workspace and BUFfer arrays                C3.2
C
CL                 4.       HOUSEKEEPING
C  COMGMA          Global Mesh Addressing                     C4.1
C  COMBTD          Block Type Data                            C4.2
C  COMPAD          Particle addressing                        C4.3
C  COMMIM          MIMD Related addressing                    C4.4
C  COMADP          ADressing Parameters                       C4.5
C  COMDIP          DImensioning Parameters                    C4.6
C
CL                 5.       I/O AND DIAGNOSTICS
C  COMOUT          OUTput variables                           C5.1
C
```

INDBLK.doc

```
C-------------------------------------------------------------------------
CL                          INDEX OF COMMON VARIABLES
C
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C
CL                  C2.1     Physical CoNstants
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMPCN/
C
C  BOLTZK          Boltzmanns constant (SI)                   R  2.1
C  CLIGHT          *Speed of light (SI)                       R  2.1
C  ELCHAG          Electron charge (SI)                       R  2.1
C  ELMASS          Electron mass (SI)                         R  2.1
C  EMU0            Permeability of free space (SI)            R  2.1
C  EOVERM          Electron charge/mass ratio                 R  2.1
C  EPS0            Free Space permittivity (SI)               R  2.1
C  PI              pi                                         R  2.1
C  Z3770           free space impedance                       R  2.1
C
CL                  C2.2     physical Iv and BC
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMIBC/
C
C  BAPLYD          *applied B field (SI)                      R  2.2
C  BCATR(MAXBCA)   **array of boundary condition attributes   RA 2.2
C  DAPLYD          applied field                              R  2.2
C  EAPLYD          *applied E field (SI)                      R  2.2
C  SPATR(MAXBCA)   *array of particle species attributes      RA 2.2
C  SURFZ           *surface impedance in Z0,s                 R  2.2
C  XLEN1(3)        *dimension of block type 1                 RA 2.2
C  XLEN2(3)        *dimension of block type 2                 RA 2.2
C  NB(3)           *no of blocks in side for NCASE=2          IA 2.2
C  NCASE           *select device initialisation case         I  2.2
C  NINIT           *select field initialisation               I  2.2
C  NODIM           *dimensionality of problem                 I  2.2
C  NOEL1(3)        *elements in block type 1 side             IA 2.2
C  NOEL2(3)        *elements in block type 2 side             IA 2.2
C  NPINIT          *select particle initialisation            I  2.2
C  NSPEC           *number of particle species                I  2.2
C  NSYMTP          field symmetry type                        I  2.2
C
CL                  C2.3     FieLD arrays
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMFLD/
C
C  B(MFADIM)       magnetic field array                       RA 2.3
C  C(MFADIM)       current array                              RA 2.3
C  D(MFADIM)       displacement field array                   RA 2.3
C  LORFBL(MAXBLK)  location of origins of field blocks        IA 2.3
C
CL                  C2.4     METric arrays
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMMET/
C
C  ECOV(MECDIM)    covariant basis vector array               RA 2.4
```

```
C  G(MGDIM)        metric tensor array                              RA 2.4
C
CL                  C2.5     PARticle array
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMPAR/
C
C  COORDS(MDPART)  particle positions and momenta                   RA 2.5
C
CL                  C3.1     NUMerical constants
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMNUM/
C
C  COUR            *max courant number                              R  3.1
C  DT              Timestep                                         R  3.1
C  SCALE(32)       scale to SI factors                              RA 3.1
C
CL                  C3.2     workspace and BUFfer arrays
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMBUF/
C
C  BAV(MFADIM)     workspace array for timecentered b               RA 3.2
C  EHWSP(MFADIM)   workspace array for E and H                      RA 3.2
C  GPATI(MDIOB)    gluepatch input buffer                           RA 3.2
C  GPATO(MDIOB)    gluepatch output buffer                          RA 3.2
C  GPATT(MDIOB)    gluepatch temp buffer                            RA 3.2
C  LPHOC(MDHOC)    List of Particle Head of Chains for block        IA 3.2
C  LPNXT(MDLNK)    particle buffer link list array                  IA 3.2
C  MBTOGP(MAXPB)   buffer to gluepatch pointer                      IA 3.2
C  MGPTOB(2,MAXPCH)
C                  gluepatch to buffer pointer                      IA 3.2
C  MPBEGI(MAXPB)   origin of patch space in input buffer            IA 3.2
C  MPBEGO(MAXPB)   origin of patch space in output buffer           IA 3.2
C  MPLENI(MAXPB)   length in words(4B) of patch input buffer        IA 3.2
C  MPLENO(MAXPB)   length in words(4B) of patch output buffer       IA 3.2
C  NPADB           pointer to first free location in buffer GPA I  3.2
C  NPINB           number of patches in buffer GPATO                I  3.2
C  NPINBI          number of patches in buffer GPATI                I  3.2
C
CL                  C4.1     Global Mesh Addressing
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C      COMMON/COMGMA/
C
C  ROTG2L(3,3,MAXBLK)
C                  block global to local coord rotation             RA 4.1
C  XYZBLK(3,4,MAXBLK)
C                  block corner global coords                       RA 4.1
C  MBKPAT(MAXBLK)  block to patch hoc table                         IA 4.1
C  MBKTYP(MAXBLK)  block to blocktype pointer                       IA 4.1
C  MEMBCA(MAXPCH)  EM patch to BC attribute table                   IA 4.1
C  MKSHFT(3)       patchkey packing code                            IA 4.1
C  MPABCA(MAXPCH)  particle patch to BC attribute table             IA 4.1
C  MPATBK(2,MAXPCH)
C                  patch to block pointer                           IA 4.1
C  MPATO(2,MAXPCH) location code of patch origin on blocks          IA 4.1
C  MPATX(2,MAXPCH) location code of patch extreme on blocks         IA 4.1
```

INDBLK.doc

```
C  MPATYP(MAXPCH)  patch type index                               IA 4.1
C  NBLOCK          *number of blocks                              I  4.1
C  NPATCH          number of patches                              I  4.1
C  NXTPAT(2,MAXPCH)
C                  link to next patch on block                    IA 4.1
C
CL                C4.2     Block Type Data
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMBTD/
C
C  LBLAS(MDBLAS)   list of block addressing structures            IA 4.2
C  LECOVA(NOECA,MDLGAD)
C                  list of ECOV Addressing                        IA 4.2
C  LGBHAD(NOGAD,MDLGAD)
C                  list of G addressing for H=G b                 IA 4.2
C  LGDEAD(NOGAD,MDLGAD)
C                  list of G addressing for E=G d                 IA 4.2
C  LOAGBH(MAXBLK)  location of GBH addr. in LGBHAD                IA 4.2
C  LOAGDE(MAXBLK)  location of GDE addr. in LGDEAD                IA 4.2
C  LOBLAS(MAXBLK)  location of origins of block addresses         IA 4.2
C  LOECOA(MAXBLK)  location of addressing in LECOVA               IA 4.2
C  LOECOV(MAXBLK)  location of origins in ECOV array              IA 4.2
C  LOOGBH(MAXBLK)  location of origins of G blocks                IA 4.2
C  LORGBH(MAXBLK)  location of origins of GBH in G array          IA 4.2
C  LORGDE(MAXBLK)  location of origins of GDE in G array          IA 4.2
C  NBTYPE          number of block types                          I  4.2
C
CL                C4.3     Particle addressing
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMPAD/
C
C  LOCOOR(MAXBLK)  Location of Origins of COORdinates             IA 4.3
C  LOPARA(MAXBLK)  Location of Origin of blocks in LPARAS         IA 4.3
C  LPARAS(MDCOA)   List of addressing for coordinates             IA 4.3
C
CL                C4.4     MIMD Related addressing
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMMIM/
C
C  MBKPES(MAXBLK)  block to processor pointer                     IA 4.4
C  MPRBLK(MAXPES)  process to block hoc table                     IA 4.4
C  MSTAT(MAXPB)    Status of message array                        IA 4.4
C  NXTBLK(MAXBLK)  next block in processor                        IA 4.4
C
CL                C4.5     ADressing Parameters
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMADP/
C
C  MBAINC          bc attribute table step                        I  4.5
C  MBCRD           block bordering depth                          I  4.5
C  MINC[illegible]  Mesh INCrement Origin in BLAS                 I  4.5
C  MINCOE          Mesh INCrement Origin in Ecov addressing       I  4.5
C  M[illegible]    Mesh INCrement Origin in [illegible] addressing I  4.5
C  MNMOM           loc of No of particle MOMentum coords in LPA   I  4.5
C  MNPOS           loc of No of particle POSition coords in LPA   I  4.5
```

```
C  MOCPS           offset for Charge Per Superparticle value      I  4.5
C  MODKEY          Mesh orthog and dimension key location         I  4.5
C  MOFSET          Mesh OFfSET loaction in BLAS                   I  4.5
C  MONEPS          offset for No of Electrons Per Superparticle   I  4.5
C  MORGTI          Mesh ORIGin of 1/T in Ecov addressing          I  4.5
C  MORIGE          Mesh ORIGin of e in Ecov addressing            I  4.5
C  MORIGG          Mesh ORIGin in G addressing                    I  4.5
C  MORIGT          Mesh ORIGin of T in Ecov addressing            I  4.5
C  MPAINC          particle attribute table step                  I  4.5
C  MPNOO           Particle NO Origin in LPARAS                   I  4.5
C  MPORO           Particle Origin table Origin in LPARAS         I  4.5
C  MSIZO           Mesh SIZe Origin in BLAS                       I  4.5
C  MSPACE          Mesh SPACE reserved location in BLAS           I  4.5
C  MSPEC           location of no of SPECies in LPARAS            I  4.5
C  MXFACE          Max number block faces (=6)                    I  4.5
C  MXPDIM          Max number of physical dimensions(=3)          I  4.5
C  NOBLAS          no of LBLAS entries per block                  I  4.5
C  NOECA           no of ECOV addressing entries per component    I  4.5
C  NOECC           no of ECOV components per block                I  4.5
C  NOGAD           no of G addressing entries per component       I  4.5
C  NOGCO           no of G components per block                   I  4.5
C  NOPARA          no of LPARAS entries per block                 I  4.5
C
CL                C4.6     DImensioning Parameters
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMDIP/
C
C  MAXBCA          max number of boundary condition attributes    I  4.6
C  MAXBLK          max number of blocks                           I  4.6
C  MAXBLT          max number of block types                      I  4.6
C  MAXPB           max number of patches in buffer                I  4.6
C  MAXPCH          max number of patches                          I  4.6
C  MAXPES          max number of processes                        I  4.6
C  MAXPOR          max number of processors                       I  4.6
C  MDBLAS          Minimum Dimension of LBLAS                     I  4.6
C  MDCOA           Dimension of particle addressing structure     I  4.6
C  MDECAD          Minimum Dimension of ECov ADdressing arrays    I  4.6
C  MDFWSP          Dimension of Field WorkSPace array             I  4.6
C  MDHOC           Dimension of particle block IO hoc array       I  4.6
C  MDIOB           Dimension of IO Buffer arrays                  I  4.6
C  MDLGAD          Minimum Dimension of G ADdressing arrays       I  4.6
C  MDLNK           Dimension of particle block link array         I  4.6
C  MDPART          Dimension of PARTicle coordinate arrays        I  4.6
C  MECDIM          ECov array DIMension                           I  4.6
C  MFADIM          Field Array DIMension                          I  4.6
C  MGDIM           Metric G array DIMension                       I  4.6
C  MXSPEC          max number of particle species                 I  4.6
C
CL                C5.1     OUTput variables
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C       COMMON/COMOUT/
C
C  WINXO           plot WINdow global X origin                    R  5.1
C  WINXX           plot WINdow global X eXtreme                   R  5.1
C  WINYO           plot WINdow global Y Origin                    R  5.1
```

```
C  WINYX          plot WINdow global Y eXtreme                  R  5.1
C  XPSMAX         Maximum x for pspace                          R  5.1
C  XPSMIN         Minimum x for pspace                          R  5.1
C  YPSMAX         Maximum y for pspace                          R  5.1
C  YPSMIN         Minimum y for pspace                          R  5.1
C  MIMDOP         **if =1, diagnosics for MIMD, =2 for XPATCH   I  5.1
C  NGMAX          *Max no of grid file frames                   I  5.1
C  NOPSEL         **Select Output Sequence                      I  5.1
C  NS1            **output every NS1 steps                      I  5.1
C  NXPTDD         **select 0.4 EXPERT diagnostic dump           I  5.1
C
```

INDVAR.doc

```
C-----------------------------------------------------------------------
CL                      ALPHABETIC INDEX OF COMMON VARIABLES
C
C VERSION 1.00.00   JWE/RWH   Culham Laboratory   May 1992
C
C  B(MFADIM)        magnetic field array                          RA 2.3
C  BAPLYD           *applied B field (SI)                         R  2.2
C  BAV(MFADIM)      workspace array for timecentered b            RA 3.2
C  BCATR(MAXBCA)    **array of boundary condition attributes      RA 2.2
C  BOLTZK           Boltzmanns constant (SI)                      R  2.1
C  C(MFADIM)        current array                                 RA 2.3
C  CLIGHT           *Speed of light (SI)                          R  2.1
C  COORDS(MDPART)   particle positions and momenta                RA 2.5
C  COUR             *max courant number                           R  3.1
C  D(MFADIM)        displacement field array                      RA 2.3
C  DAPLYD           applied field                                 R  2.2
C  DT               Timestep                                      R  3.1
C  EAPLYD           *applied E field (SI)                         R  2.2
C  ECOV(MECDIM)     covariant basis vector array                  RA 2.4
C  EHWSP(MFADIM)    workspace array for E and H                   RA 3.2
C  ELCHAG           Electron charge (SI)                          R  2.1
C  ELMASS           Electron mass (SI)                            R  2.1
C  EMU0             Permeability of free space (SI)               R  2.1
C  EOVERM           Electron charge/mass ratio                    R  2.1
C  EPS0             Free Space permittivity (SI)                  R  2.1
C  G(MGDIM)         metric tensor array                           RA 2.4
C  GPATI(MDIOB)     gluepatch input buffer                        RA 3.2
C  GPATO(MDIOB)     gluepatch output buffer                       RA 3.2
C  GPATT(MDIOB)     gluepatch temp buffer                         RA 3.2
C  LBLAS(MDBLAS)    list of block addressing structures           IA 4.2
C  LECOVA(NOECA,MDLGAD)
C                   list of ECOV Addressing                       IA 4.2
C  LGBHAD(NOGAD,MDLGAD)
C                   list of G addressing for H=G b                IA 4.2
C  LGDEAD(NOGAD,MDLGAD)
C                   list of G addressing for E=G d                IA 4.2
C  LOAGBH(MAXBLK)   location of GBH addr. in LGBHAD               IA 4.2
C  LOAGDE(MAXBLK)   location of GDE addr. in LGDEAD               IA 4.2
C  LOBLAS(MAXBLK)   location of origins of block addresses        IA 4.2
C  LOCOOR(MAXBLK)   Location of Origins of COORdinates            IA 4.3
C  LOECOA(MAXBLK)   location of addressing in LECOVA              IA 4.2
C  LOECOV(MAXBLK)   location of origins in ECOV array             IA 4.2
C  LOOGBH(MAXBLK)   location of origins of G blocks               IA 4.2
C  LOPARA(MAXBLK)   Location of Origin of blocks in LPARAS        IA 4.3
C  LORFBL(MAXBLK)   location of origins of field blocks           IA 2.3
C  LORGBH(MAXBLK)   location of origins of GBH in G array         IA 4.2
C  LORGDE(MAXBLK)   location of origins of GDE in G array         IA 4.2
C  LPARAS(MDCOA)    List of addressing for coordinates            IA 4.3
C  LPHOC(MDHOC)     List of Particle Head of Chains for block     IA 3.2
C  LPNXT(MDLNK)     particle buffer link list array               IA 3.2
C  MAXBCA           max number of boundary condition attributes   I  4.6
C  MAXBLK           max number of blocks                          I  4.6
C  [illegible]      max number of [illegible] types               [illegible] 4.[illegible]
C  MAXPB            max number of patches in buffer               [illegible] 4.[illegible]
C  MAXPCH           max number of patches                         I  4.6
C  MAXPES           max number of processes                       I  4.6
C  MAXPOR           max number of processors                      I  4.6
C  MBAINC           bc attribute table step                       I  4.5
C  MBKPAT(MAXBLK)   block to patch hoc table                      IA 4.1
C  MBKPES(MAXBLK)   block to processor pointer                    IA 4.4
C  MBKTYP(MAXBLK)   block to blocktype pointer                    IA 4.1
C  MBORD            block bordering depth                         I  4.5
C  MBTOGP(MAXPB)    buffer to gluepatch pointer                   IA 3.2
C  MDBLAS           Minimum Dimension of LBLAS                    I  4.6
C  MDCOA            Dimension of particle addressing structure    I  4.6
C  MDECAD           Minimum Dimension of ECov ADdressing arrays   I  4.6
C  MDFWSP           Dimension of Field WorkSPace array            I  4.6
C  MDHOC            Dimension of particle block IO hoc array      I  4.6
C  MDIOB            Dimension of IO Buffer arrays                 I  4.6
C  MDLGAD           Minimum Dimension of G ADdressing arrays      I  4.6
C  MDLNK            Dimension of particle block link array        I  4.6
C  MDPART           Dimension of PARTicle coordinate arrays       I  4.6
C  MECDIM           ECov array DIMension                          I  4.6
C  MEMBCA(MAXPCH)   EM patch to BC attribute table                IA 4.1
C  MFADIM           Field Array DIMension                         I  4.6
C  MGDIM            Metric G array DIMension                      I  4.6
C  MGPTOB(2,MAXPCH)
C                   gluepatch to buffer pointer                   IA 3.2
C  MIMDOP           **if =1, diagnosics for MIMD, =2 for XPATCH   I  5.1
C  MINCO            Mesh INCrement Origin in BLAS                 I  4.5
C  MINCOE           Mesh INCrement Origin in Ecov addressing      I  4.5
C  MINCOG           Mesh INCrement Origin in G addressing         I  4.5
C  MKSHFT(3)        patchkey packing code                         IA 4.1
C  MNMOM            loc of No of particle MOMentum coords in LPA  I  4.5
C  MNPOS            loc of No of particle POSition coords in LPA  I  4.5
C  MOCPS            offset for Charge Per Superparticle value     I  4.5
C  MODKEY           Mesh orthog and dimension key location        I  4.5
C  MOFSET           Mesh OFfSET loaction in BLAS                  I  4.5
C  MONEPS           offset for No of Electrons Per Superparticle  I  4.5
C  MORGTI           Mesh ORIGin of 1/T in Ecov addressing         I  4.5
C  MORIGE           Mesh ORIGin of e in Ecov addressing           I  4.5
C  MORIGG           Mesh ORIGin in G addressing                   I  4.5
C  MORIGT           Mesh ORIGin of T in Ecov addressing           I  4.5
C  MPABCA(MAXPCH)   particle patch to BC attribute table          IA 4.1
C  MPAINC           particle attribute table step                 I  4.5
C  MPATBK(2,MAXPCH)
C                   patch to block pointer                        IA 4.1
C  MPATO(2,MAXPCH)  location code of patch origin on blocks       IA 4.1
C  MPATX(2,MAXPCH)  location code of patch extreme on blocks      IA 4.1
C  MPATYP(MAXPCH)   patch type index                              IA 4.1
C  MPBEGI(MAXPB)    origin of patch space in input buffer         IA 3.2
C  MPBEGO(MAXPB)    origin of patch space in output buffer        IA 3.2
C  MPLENI(MAXPB)    length in words(4B) of patch input buffer     IA 3.2
C  MPLENO(MAXPB)    length in words(4B) of patch output buffer    IA 3.2
C  MPNOO            Particle NO Origin in LPARAS                  I  4.5
C  MPORO            Particle Origin table Origin in LPARAS        I  4.5
C  MPRBLK(MAXPES)   process to block hoc table                    IA 4.4
C  [illegible]      [illegible] BLAS                              [illegible]
C  MSPACE           Mesh SPACE reserved location in BLAS          I  4.5
C  MSPEC            location of no of SPECies in LPARAS           I  4.5
```

```
C  MSTAT(MAXPB)    Status of message array                      IA 4.4
C  MXFACE          Max number block faces (=6)                  I  4.5
C  MXPDIM          Max number of physical dimensions(=3)        I  4.5
C  MXSPEC          max number of particle species               I  4.6
C  NB(3)           *no of blocks in side for NCASE=2            IA 2.2
C  NBLOCK          *number of blocks                            I  4.1
C  NBTYPE          number of block types                        I  4.2
C  NCASE           *select device initialisation case           I  2.2
C  NGMAX           *Max no of grid file frames                  I  5.1
C  NINIT           *select field initialisation                 I  2.2
C  NOBLAS          no of LBLAS entries per block                I  4.5
C  NODIM           *dimensionality of problem                   I  2.2
C  NOECA           no of ECOV addressing entries per component  I  4.5
C  NOECC           no of ECOV components per block              I  4.5
C  NOEL1(3)        *elements in block type 1 side               IA 2.2
C  NOEL2(3)        *elements in block type 2 side               IA 2.2
C  NOGAD           no of G addressing entries per component     I  4.5
C  NOGCO           no of G components per block                 I  4.5
C  NOPARA          no of LPARAS entries per block               I  4.5
C  NOPSEL          **Select Output Sequence                     I  5.1
C  NPADB           pointer to first free location in buffer GPA I  3.2
C  NPATCH          number of patches                            I  4.1
C  NPINB           number of patches in buffer GPATO            I  3.2
C  NPINBI          number of patches in buffer GPATI            I  3.2
C  NPINIT          *select particle initialisation              I  2.2
C  NS1             **output every NS1 steps                     I  5.1
C  NSPEC           *number of particle species                  I  2.2
C  NSYMTP          field symmetry type                          I  2.2
C  NXPTDD          **select 0.4 EXPERT diagnostic dump          I  5.1
C  NXTBLK(MAXBLK)  next block in processor                      IA 4.4
C  NXTPAT(2,MAXPCH)
C                  link to next patch on block                  IA 4.1
C  PI              pi                                           R  2.1
C  ROTG2L(3,3,MAXBLK)
C                  block global to local coord rotation         RA 4.1
C  SCALE(32)       scale to SI factors                          RA 3.1
C  SPATR(MAXBCA)   *array of particle species attributes        RA 2.2
C  SURFZ           *surface impedance in Z0,s                   R  2.2
C  WINXO           plot WINdow global X Origin                  R  5.1
C  WINXX           plot WINdow global X eXtreme                 R  5.1
C  WINYO           plot WINdow global Y Origin                  R  5.1
C  WINYX           plot WINdow global Y eXtreme                 R  5.1
C  XLEN1(3)        *dimension of block type 1                   RA 2.2
C  XLEN2(3)        *dimension of block type 2                   RA 2.2
C  XPSMAX          Maximum x for pspace                         R  5.1
C  XPSMIN          Minimum x for pspace                         R  5.1
C  XYZBLK(3,4,MAXBLK)
C                  block corner global coords                   RA 4.1
C  YPSMAX          Maximum y for pspace                         R  5.1
C  YPSMIN          Minimum y for pspace                         R  5.1
C  Z377O           free space impedance                         R  2.1
C
```

D Annex 3: Running *mimdpic* on the iPSC

```
 --------------------------------------
|            LPM2  BENCHMARK           |
|            ---------------           |
|     (Intel native communications)    |
|                                      |
|   Roger Hockney and James Eastwood   |
|              May 1993                |
|                                      |
 --------------------------------------
```

First Read the directory picpac2.d (7.5MB) from the Sun tar cartridge tape to the computer file system, with a UNIX command like (use 150MB drive):

```
                tar xvf /dev/rst0  picpac2.d       ...  on a SUN
or              tar xvf /dev/rmt0  picpac2.d       ... on an IBM
```

The following is the ReadMe file in directory picpac2.d

DESCRIPTION

The Local Particle Mesh (LPM2) Benchmark was written for the USAF to measure the parallelization properties of the new Electro-Magnetic PIC code for the simulation of MILO type devices on massively parallel computers (MPPs), such as the Intel iPSC/860 and Intel Paragon.

This version uses the Intel native communication library (csend, crecv etc). Conversion to other systems can be made by placing appropriate alternative communication calls in subroutines NODASG (c1s12z.f), XPATCH (c2s31z.f) and benctl.f only.

The program can be run with or without graphics. Simple graphics is provided to use as a demonstration and as a check on the validity of the calculation. The geometry of the device is drawn before the timing period, with the region computed by each process shown in a different color (although there are only four colors which are cycled through). Only the boundaries of the blocks are drawn, because the mesh itself is too fine. After the end of the timing interval, all the particles are drawn (in the appropriate color for each process) on top of the device diagram. Printed output of the benchmark timing data and performance is sent to the screen and also to an output file.

The device is made up of blocks, and parallelization is achieved by assigning blocks to processes within the program (NODASG and SETDMP), and external to the program by assigning processes to processors (or nodes). The program is designed to allow dynamic allocation of blocks to processes, but the benchmark program uses a fixed allocation. Allocation of processes to processors, depends on facilities provided by the parallel computer. In this version for the Intel computers, we assume one process to each processor.

Interfacing the graphics to a new computer system is tricky, and we recommend that benchmark times be made without the graphics display. The program has in internal check on the validity of the calculation, based on comparing the total number of particles at the end of the benchmark run with a reference number (obtained from a valid one processor calculation). If these numbers agree to better than 10%, then the calculation is regarded as valid. Exact agreement cannot be expected because of the use of random numbers for particle injection. There seems to be no way of ensuring that a multi-process run uses the same random numbers for the same purposes as a single-process run, therefore we can only expect approximate agreement on a statistical basis.

CASES

This initial benchmark tape provides for two cases:

case1: a small MILO problem with 5 cavities and 12 blocks and about 100 particles, which has been used as a test calculation during program development. It is small enough to run on any workstation and give a reasonable real-time display when run on one processor. It is too small, however, to make sensible use of a massively parallel computer, and the speedup behavior will be disappointing.

case2: a moderately sized problem with 31 cavities and 64 blocks, and about 12000 particles, which should be big enough to show reasonable speedup behavior on a massively parallel computer.

a further case is planned:

case3: (to be created after further knowledge of and some experience with the USAF MPP) a massive problem tuned to make the most of the massively parallel computer being used.

DIRECTORIES

The following directories are provided on the benchmark tape, only the first is needed for benchmarking

picpac2.d (7.4MB)	LPM2 benchmarking directory, for iPSC without graphics

The following may be used to demonstrate the program on a Sun

picsim.d (8.6MB)	LPM2 benchmarking directory, for Sun simulator and on-line (xgenie) graphics

the following directories are not needed for elementary benchmarking and can usually be ignored. They may occasionally be needed to get out of trouble:

ipscsim.d (4.5MB)	iPSC simulator
lpmlibm.d (2.1MB)	LPM1 benchmark
XGENIE (1.6MB)	source for xgenie graphics (not multiplexed)
XGENIE.NEW(1.3MB)	ditto for multiplexed graphics

OPERATING INSTRUCTIONS (benchmark without graphics - UNIX commands)

The files on tape are setup for benchmark runs on an Intel iPSC/860, which may be run as follows:

```
(1)    cd picpac2.d                 = go to benchmark directory
(2)    cp case1.dat input.dat       = copy case1 1-proc data to input.dat
                                    = which is used as a working input file
(3)    getcube -t1                  = get a cube of 1-processor
(4)    sh host.sh                   = load and start running
(5)            = results to screen and file o_case1p1:1

(6)    vi input.dat       = edit line-2 to set NPRES=2, for 2 processes
                            change last character to number of processes
(7)    getcube -t2        = -t2 gets 2 nodes
(8)    sh host
```

```
(9)             - results to screen and files:
                          o_case1p2:1 (process 1)
                          o_case1p2:2 (process 2)

(10)   repeat steps (6) to (9) for at least 10 different numbers of processes
       (=NPRES), roughly equally spaced logarithmically:
            e.g.  3,4,5,6,8,10,12,16,20,24,32,40,50,64  on 64-node iPSC
       The separate output files for each process are generated automatically
                              o_case<n>p<m>:<r>
            output for process <r> from <m>-process run of case<n>

(11)   cp case2.dat input.dat           = bring-in case2 data for 1-proc run
(12)   repeat steps (3) to (10) using case2 data
```

Example output can be found in: picpac2.d/pac_o_*   and picsim.d/o_*

BENCHMARKING COMPLETE (case1 and case2)

RECOMPILATION

The benchmark directory contains executables which have run on an iPSC/860, and should be able to be used as described above. However if they do not work, or if it is desired to recompile and link at a different level of optimization, new executables can be produced as follows, using the UNIX makefile facility. The following makefiles are provided:

```
           makefile          = the working makefile
           makefile.iPSC     = to make executables for an iPSC/860
           makefile.sim      = to make executables to use with the Intel
                               iPSC simulator, running on a SUN workstation
```

to use any of these:

```
           cp makefile.iPSC makefile   = setup for real iPSC
           make                        = recompile and link all changed files
                                         and produce executable called 'node'
```

The instructions above run a shell script called 'host.sh' to load the executable 'node' onto all nodes and start execution. After all nodes have finished the cube is released.

USING GRAPHICS

Two forms of graphics are possible:

(1) GHOST interface - Off-line graphics is provided if the Culham laboratory Ghost graphics library is available. Each node can be made to write a separate file of graphics instructions that can be subsequently processed and viewed, using standard Ghost interactive facilities.

```
      (1.1)   cp .FOR_O_LIS.ghost  .FOR_O_LIS.inc     removes ghost dummies
                                                      in cpslz.f
      (1.2)   cp lib_lis.ghost  lib_lis.inc           include ghost library
      (1.3)   cp makefile.iPSC  makefile              to use real iPSC
     or (1.3a)  cp makefile.sim  makefile             to use simulator
      (1.4)   make                                    make new 'node'
      (1.5)   operate as benchmark program above
```

The graphics output files generated are:

g_case<n>p<m>:<r>

ghost graphics grid file for process <r> from <m>-process run of case<n>

(2) XGENIE interface - On-line graphics is provided via the xgenie daemon which is attached to the running host program via a UNIX pipe. All nodes write coded graphics instructions to the host program using standard Intel csend/crecv instructions. After loading the node program onto the iPSC/860 nodes, the host program goes into an endless loop reading coded graphics instructions from the nodes, and sending them over the pipe to xgenie, which plots them in an X-window, previously opened by the host. A control-C ends the host program when required. This also kills the nodes (usually!). The xgenie on-line graphics has been setup to run with the SUN simulator in directory:

picsim.d

This may be used to demonstrate the paralellization of the program, but cannot be used to obtain timing or performance numbers.

```
 (2.1)    cp .FOR_O_LIS.noghost  .FOR_O_LIS.inc   inserts ghost dummies
 (2.2)    cp lib_lis.noghost  lib_lis.inc         no libraries required
 (2.3)    cp host.f.graf  host.f                  host program + grafloop
 (2.4)    cp makefile.iPSC  makefile              to use real iPSC
or (2.4a)  cp makefile.sim  makefile              to use simulator
 (2.5)    make                                    make node program
 (2.6)    vi xgenie.h    line 81 put correct directory path into
                         definition of DefaultServerPath to reach
                         picsim.d/xgenie. Use absolute path from root.
 (2.7)    make xgenieIF                           compile xgenie FORTRAN
                                                  interface routines
 (2.8)    make xgenie                             make xgenie daemon
 (2.9)    make host                               make host program
```

The program must be run from the host as follows, e.g.:

```
 (2.10) cp case1.dat input.dat      setup test input
 (2.11) vi input.dat                edit line-2 last character to 4
 (2.11) getcube -t4                 get 4 nodes
 (2.12) host                        run host executable on host
           X-window appears and device is drawn
              100 step benchmark run is done
           Final particle distribution displayed
 (2.13) kill job with control-C
```

If particle trajectories are required as computation takes place, uncomment line 155 (CALL SCAPLT) in c3s1.f (OUTPUT), and type 'make'.